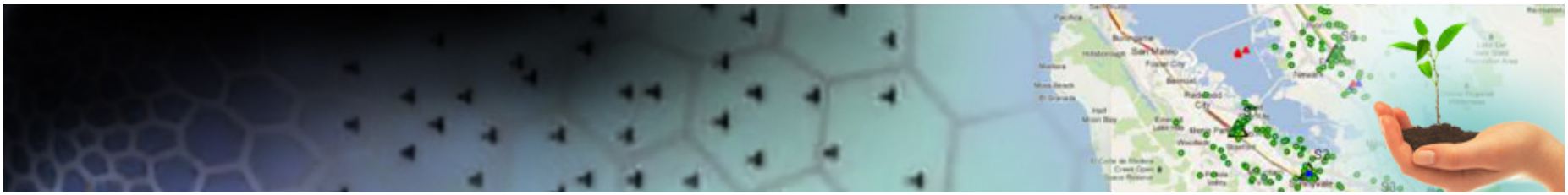
Intelligent Hardware-Enabled Sensor and Software Safety and Health Management for Autonomous UAS

Kristin Yvonne Rozier Johann Schumann Corey Ippolito



NASA Aeronautics Research Institute
Seedling Phase I Final Deliverable

December 4, 2014

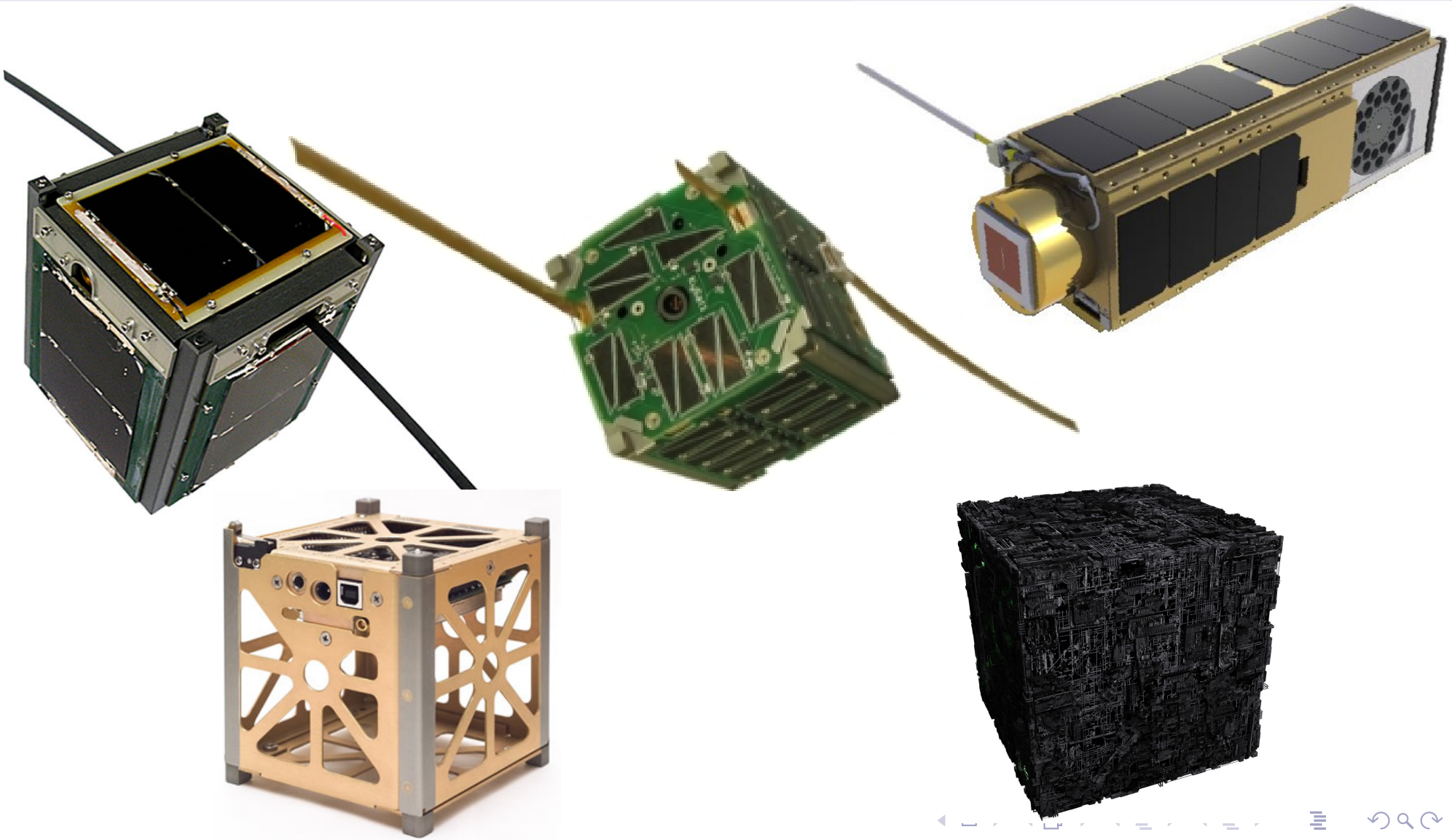
California Wild Fires

Hazardous environments

Hazardous environments

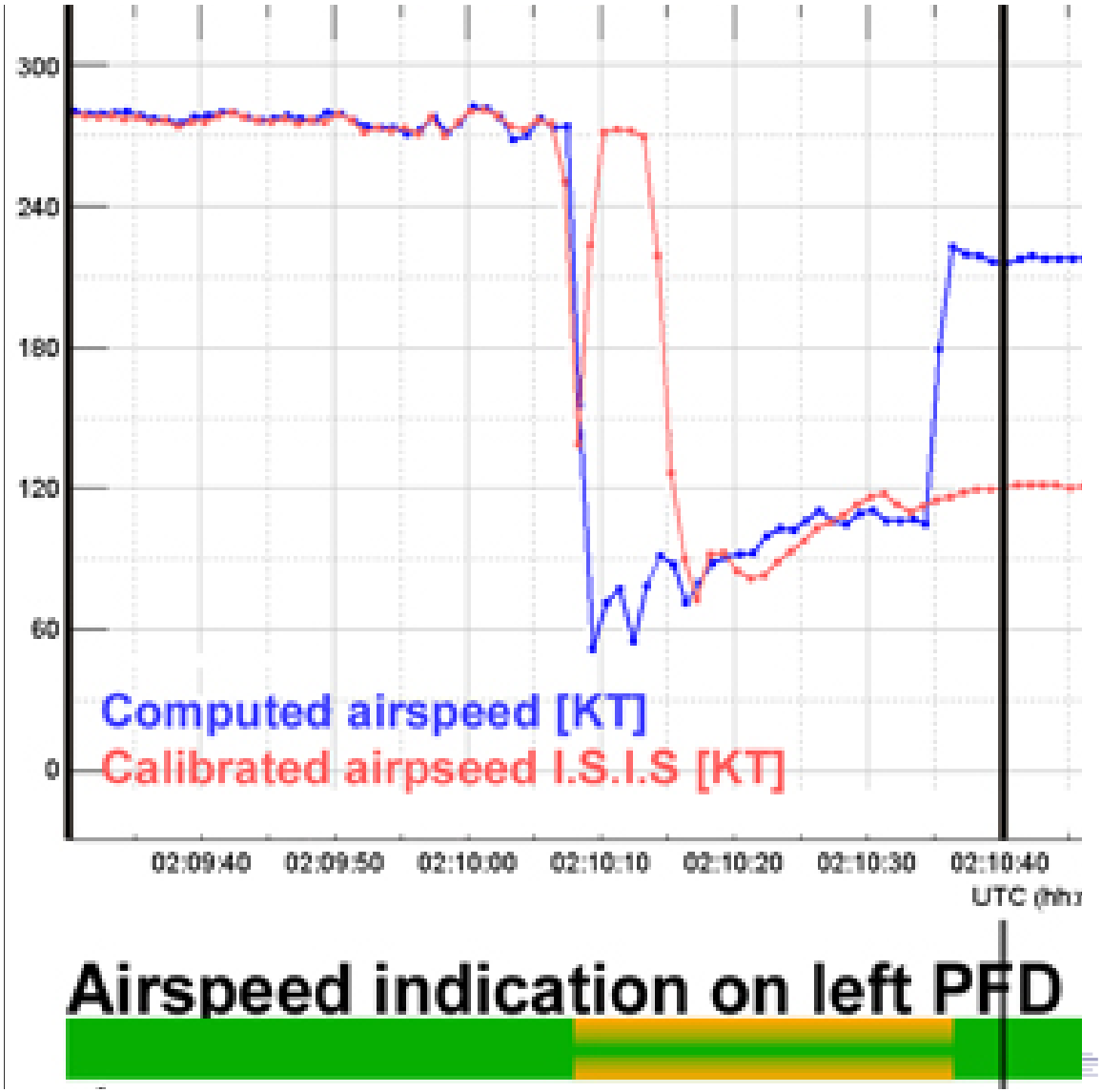
Hazardous environments

Even outer space...

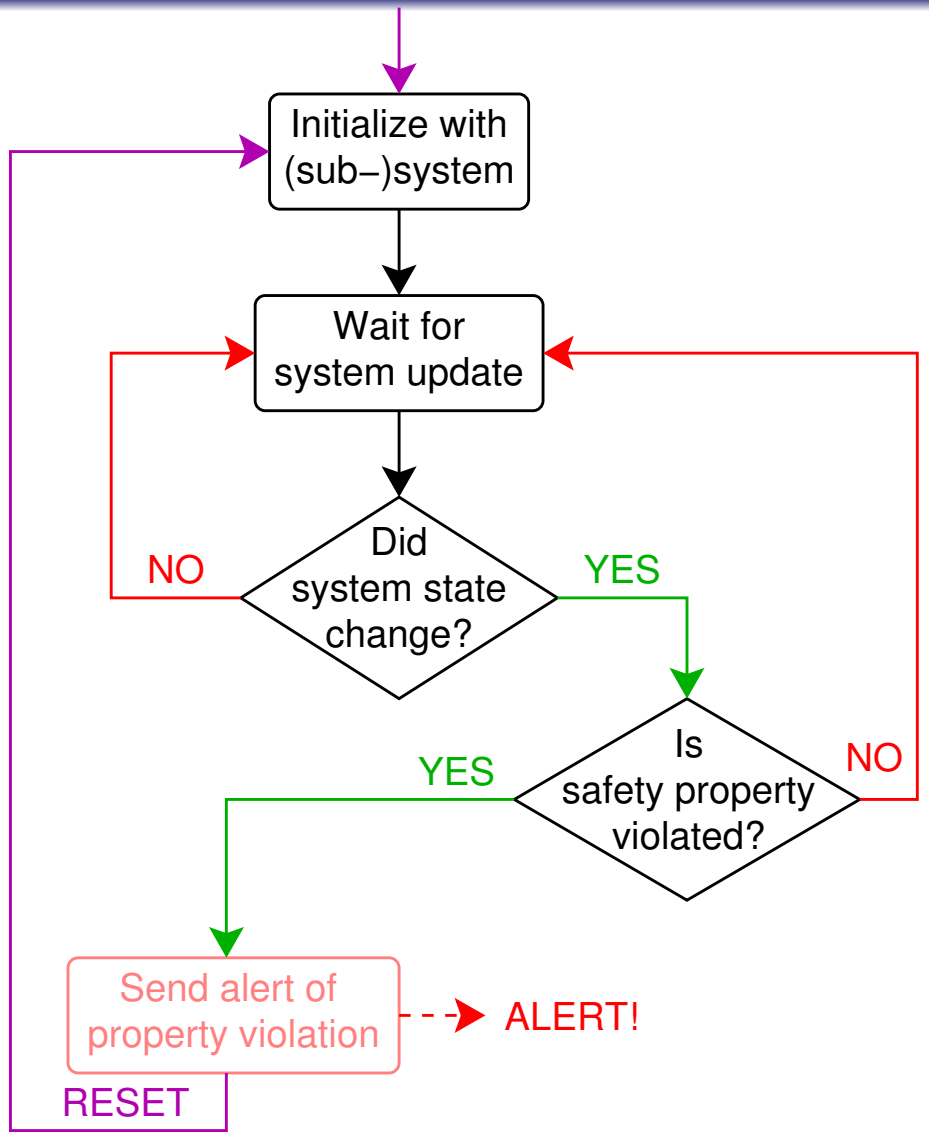


A380: Engine Exploded

AF447 Airspeed Measurements



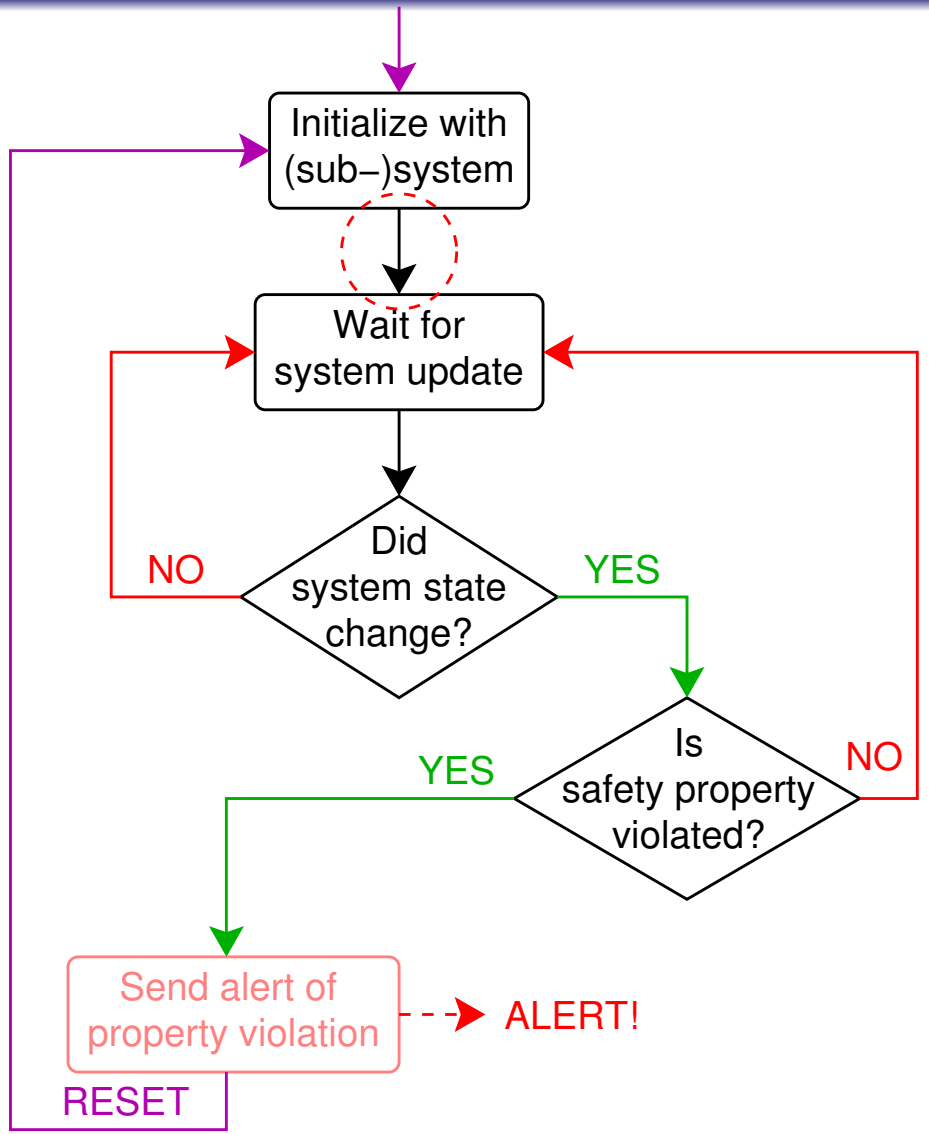
Runtime Monitoring



Research: state-of-the-art

- can check complex temporal properties *very* efficiently
- low overhead
- real-time
- enables resets, exits from infinite loops, etc.

Runtime Monitoring



Research: state-of-the-art

- can check complex temporal properties *very* efficiently
- low overhead
- real-time
- enables resets, exits from infinite loops, etc.
- requires instrumentation to send state variables to monitor

Previous Approaches: Runtime Monitoring

- report property failure
- instrument software (or hardware)
  - loses flight-certification
  - alter the original timing behavior
- utilize advanced resources
  - require a powerful computer
  - use powerful database systems
  - high overhead
  - hardware monitors: resynthesize monitors from scratch or exclude checking real-time properties
- unintuitive specification language
- report only the outcomes of specifications

Previous Approaches: Runtime Monitoring

- report property failure
- instrument software (or hardware)
  - loses flight-certification
  - alter the original timing behavior
- utilize advanced resources
  - require a powerful computer
  - use powerful database systems
  - high overhead
  - hardware monitors: resynthesize monitors from scratch or exclude checking real-time properties
- unintuitive specification language
- report only the outcomes of specifications

Requirements: Realizability

Realizability:

- plug-and-play
- easy, *expressive* specification language
- *generic* interface to connect to a wide variety of systems
- able to *adapt* to new specifications without a lengthy re-compilation
- able to efficiently monitor different requirements during *different mission stages*: {takeoff, approach, measurement, return}
- able to run on standardized components *without interfering with flight certifiability*

Requirements: Responsiveness

Responsiveness:

- *continuously monitor* the system
- *detect deviations* from the monitored specifications within a tight and a priori known time bound
- *enable mitigation* or rescue measures
  - return to base for repair
  - a controlled emergency landing to avoid damage on the ground
- report *intermediate status* and satisfaction of timed requirements as early as possible (for decision-making)

Requirements: Unobtrusiveness

UNOBTRUSIVENESS:
(not alter crucial properties of the system)

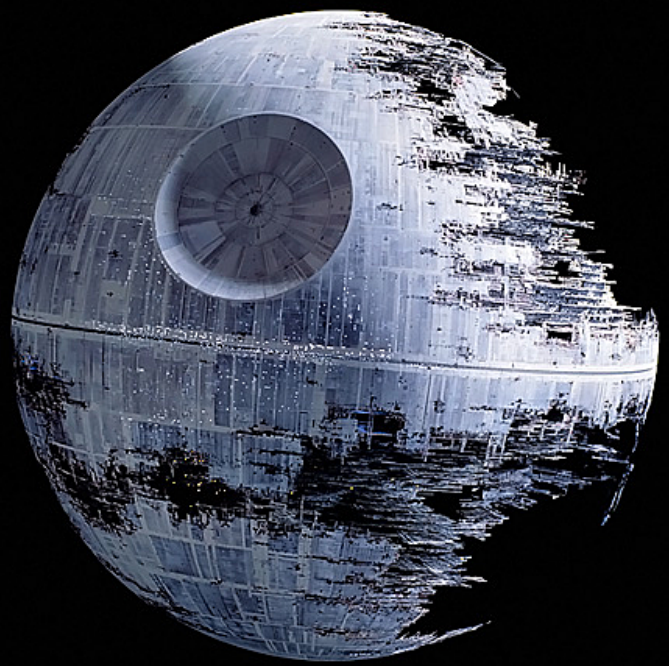
- *functionality*: not change behavior
- *certifiability*: avoid re-certification of flight software/hardware ($\Rightarrow$ read-only access)
- *timing*: not interfere with timing guarantees
- *tolerances*: not violate size, weight, power, or telemetry bandwidth constraints
- *cost*: use commercial-off-the-shelf (COTS) components
- *tight time and budget constraints*

Requirements

Adding currently available runtime monitoring capabilities to the UAS would change its flight certification.

"Losing flight certification is like moving over to the dark side: once you go there you can never come back."

— Doug McKinnon,
NASA's UAS Crew Chief



Satisfying Requirements

real-**t**ime
RESPONSIVE
REALIZABLE
UNOBTRUSIVE
Unit

rt-R2U2

Satisfying Requirements

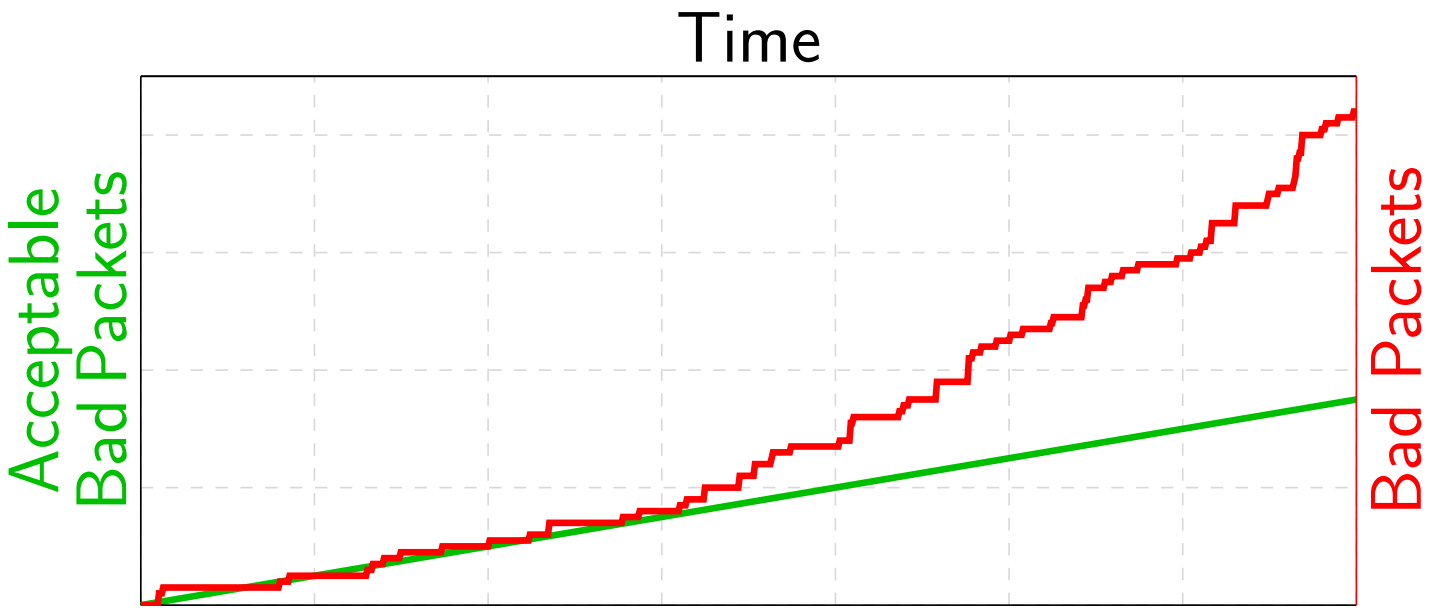
real-time
RESPONSIVE
REALIZABLE
UNOBTRUSIVE
Unit

rt-R2U2

... So how do we do that?

Fluxgate Magnetometer: Impact

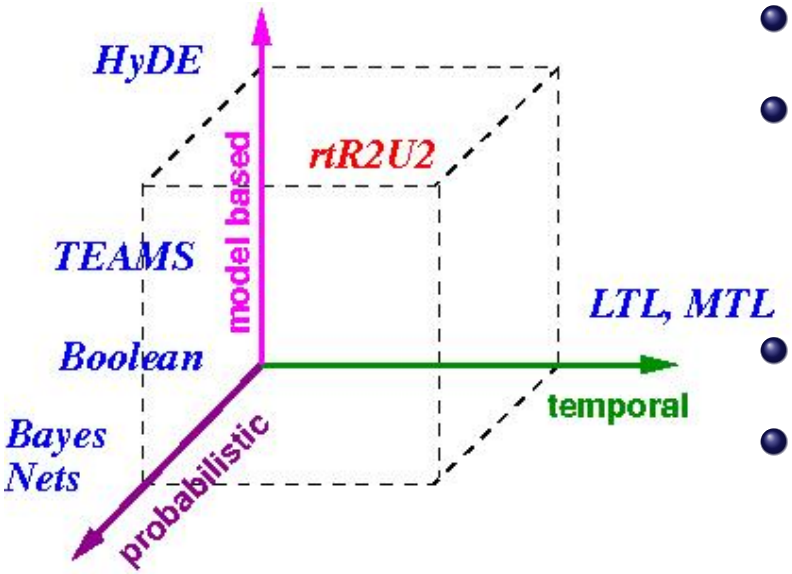
rt-R2U2 Finds Fluxgate Magnetometer Buffer Overflow



http://www.usgs.gov/blogs/surprisevalley/

Fault Detection and Monitoring

Any diagnosis system works with an *abstracted* model of the actual system

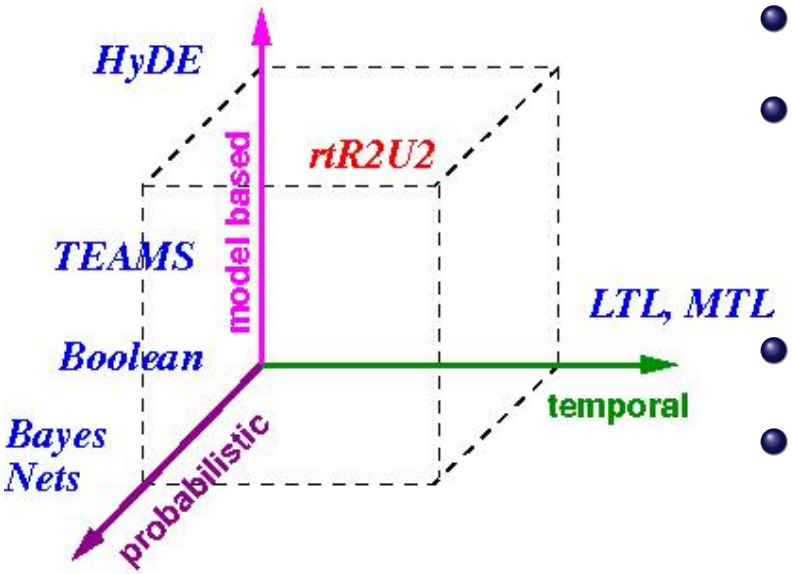


Typical Abstraction Dimensions

- Boolean conditions: "if-then-else" rules
- model-based: use hierarchical, multi-signal reachability (e.g., TEAMS) or simplified dynamic models (HyDE)
- temporal: use temporal logic
- probabilistic: use BN, or Fuzzy, or Neural Networks

Fault Detection and Monitoring

Any diagnosis system works with an *abstracted* model of the actual system



Typical Abstraction Dimensions

- Boolean conditions: "if-then-else" rules
- model-based: use hierarchical, multi-signal reachability (e.g., TEAMS) or simplified dynamic models (HyDE)
- temporal: use temporal logic
- probabilistic: use BN, or Fuzzy, or Neural Networks
- rtR2U2 combines model-based, temporal, and probabilistic paradigms for convenient modeling and high expressiveness

Runtime monitors don't make decisions.

Runtime monitors don't make decisions.

Dynamic Bayesian Networks are too complex.

Runtime monitors don't make decisions.

Dynamic Bayesian Networks are too complex.

How do we satisfy our requirements?

Runtime monitors don't make decisions.

Dynamic Bayesian Networks are too complex.

How do we satisfy our requirements?

Where do we satisfy our requirements?

Runtime monitors don't make decisions.

Dynamic Bayesian Networks are too complex.

How do we satisfy our requirements?

Where do we satisfy our requirements?

We can combine these two together to solve our problem!

Runtime monitors don't make decisions.

Dynamic Bayesian Networks are too complex.

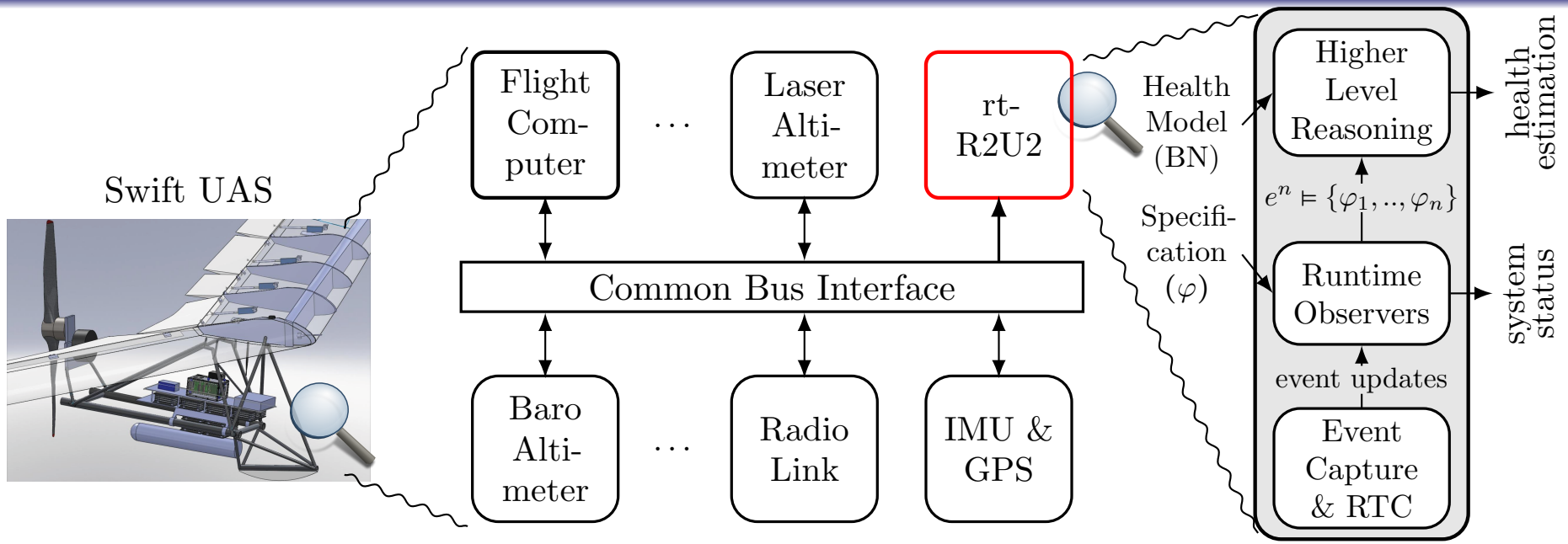
How do we satisfy our requirements?

Where do we satisfy our requirements?

We can combine these two together to solve our problem!

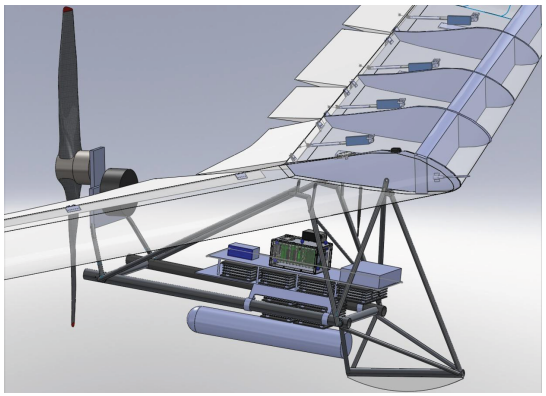
On-board a spare FPGA!

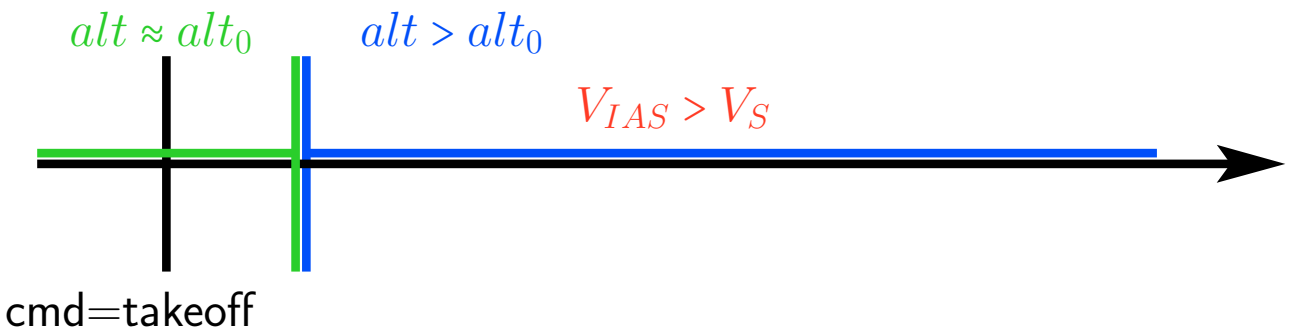
rt-R2U2



- synthesis and integration of new paired synchronous/asynchronous observer framework
- real-time, REALIZABLE, RESPONSIVE, UNOBTRUSIVE
- enables discrete Bayesian Network-based reasoning
- modular hardware implementation: FPGA
- execution of a proof-of-concept

Runtime Monitor Specifications for the Swift UAS





Whenever the Swift UAS is in the air, its indicated airspeed (V_{IAS}) must be greater than its stall speed V_S. The UAS is considered to be air-bound when its altitude *alt* is greater than that of the runway alt_0.

Asynchronous Observers (aka event-triggered)

- *evaluate with every new input*
- 2-valued output: {true; false}
- resolve specification *as early as possible* (a priori known time)

Synchronous Observers (aka time-triggered)

- update continuously
- 3-valued output: {true; false; maybe}
- small hardware footprints

Synchronous observers update at every tick of the system clock
. . . enabling probabilistic system diagnosis!

Runtime Monitoring for a UAS

After receiving a command (cmd) for takeoff, the UAS must reach an altitude of 600ft within 60 seconds.

Runtime Monitoring for a UAS

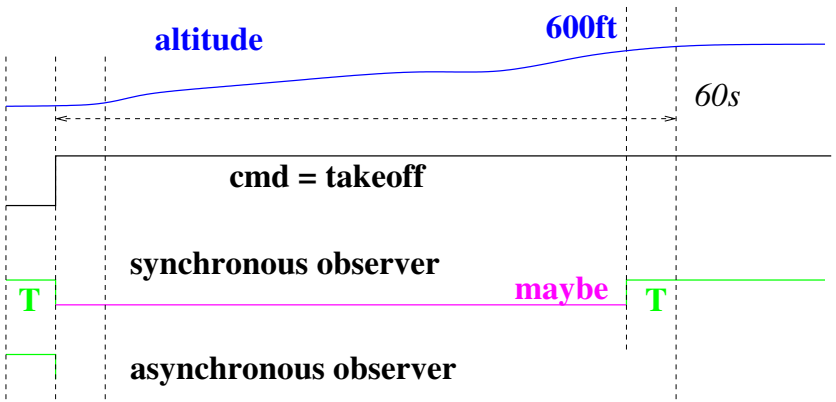
After receiving a command (cmd) for takeoff, the UAS must reach an altitude of 600ft within 60 seconds.

$$\square((\text{cmd} == \text{takeoff}) \rightarrow \Diamond_{[0,60s]}(alt \geq 600\ \text{ft}))$$

Runtime Monitoring for a UAS

After receiving a command (cmd) for takeoff, the UAS must reach an altitude of 600ft within 60 seconds.

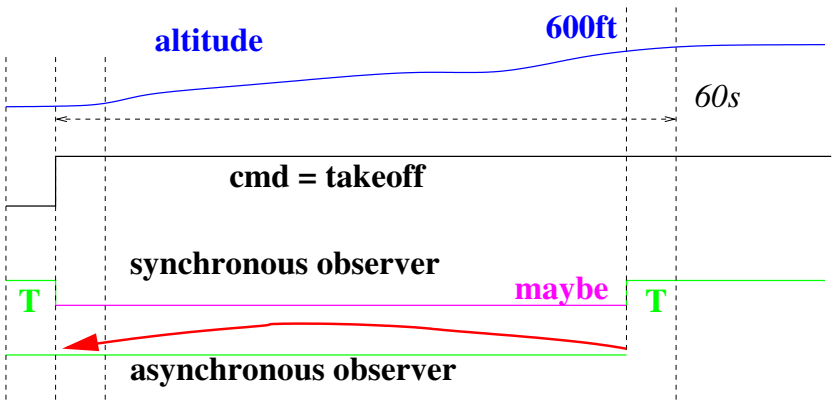
$$\Box((\text{cmd} == \text{takeoff}) \rightarrow \Diamond_{[0,60s]}(alt \geq 600\ \text{ft}))$$



Runtime Monitoring for a UAS

After receiving a command (cmd) for takeoff, the UAS must reach an altitude of 600ft within 60 seconds.

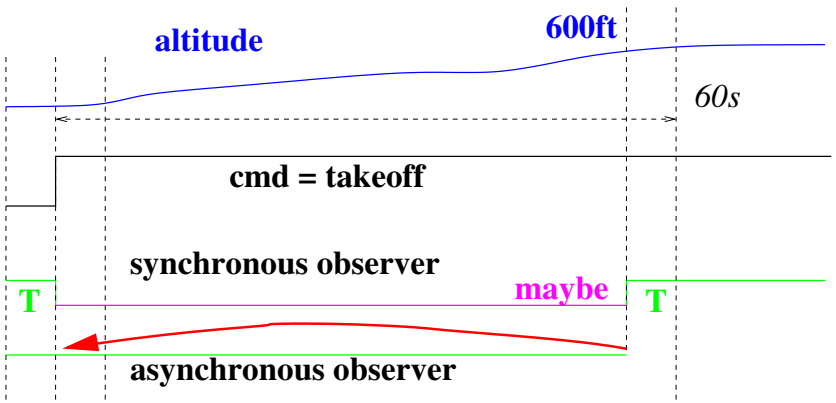
$$\Box((\text{cmd} == \text{takeoff}) \rightarrow \Diamond_{[0,60s]}(alt \geq 600\ \text{ft}))$$

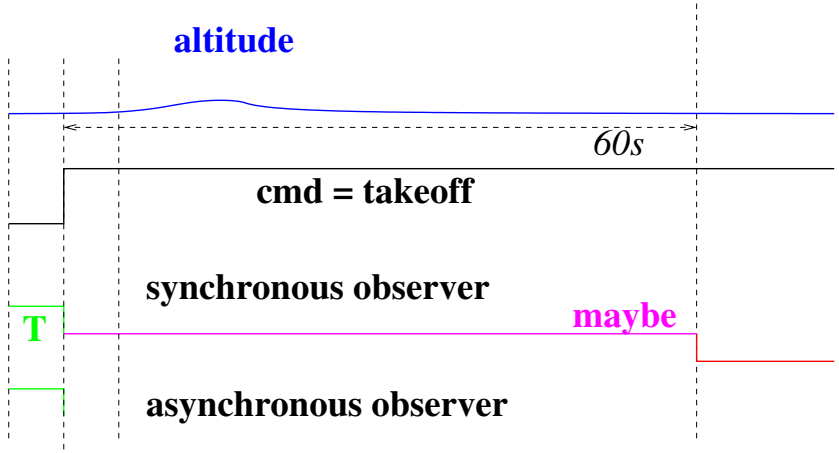


Runtime Monitoring for a UAS

After receiving a command (cmd) for takeoff, the UAS must reach an altitude of 600ft within 60 seconds.

$$\Box((\text{cmd} == \text{takeoff}) \rightarrow \Diamond_{[0,60s]}(alt \geq 600\ \text{ft}))$$

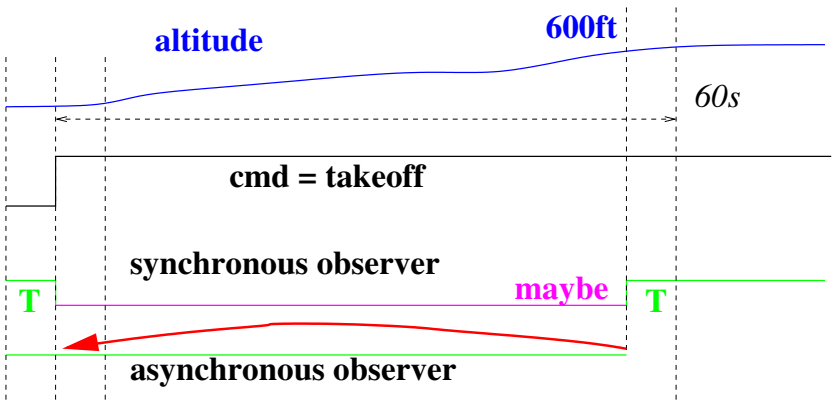


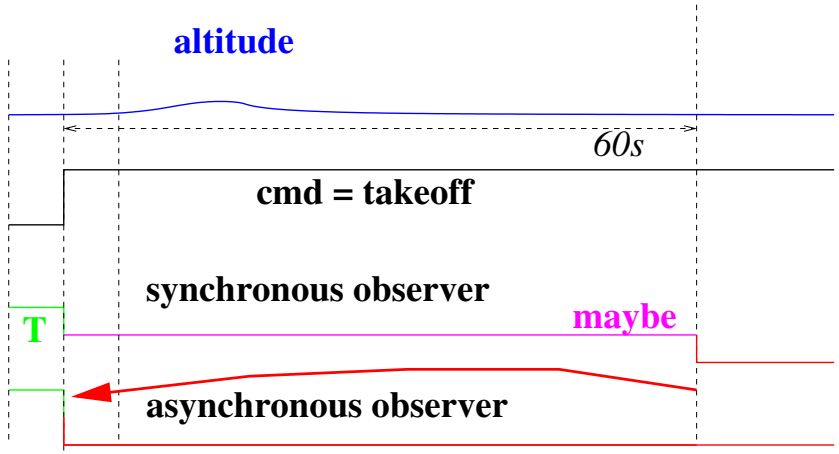


Runtime Monitoring for a UAS

After receiving a command (cmd) for takeoff, the UAS must reach an altitude of 600ft within 60 seconds.

$$\Box((\text{cmd} == \text{takeoff}) \rightarrow \Diamond_{[0,60s]}(alt \geq 600\ \text{ft}))$$

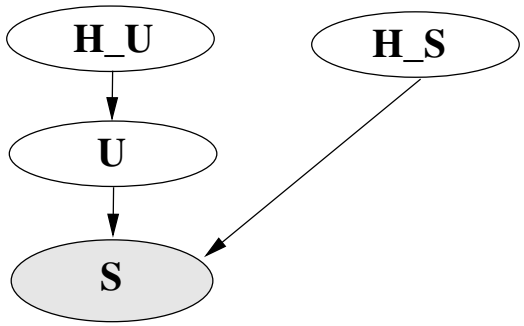




The maybe can be used by the BN reasoner for disambiguation

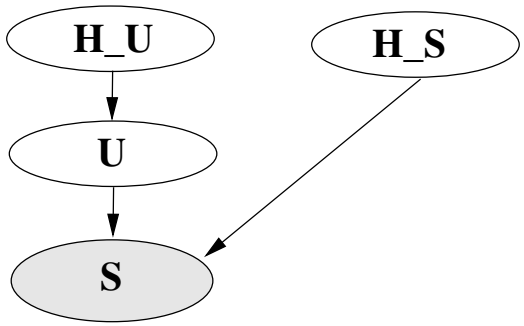
Bayesian Reasoning

- Bayesian models for diagnostic reasoning and health management are well established
- Our Bayesian Networks (BNs) contain
  - (observable) sensor nodes S, which can be noisy of fail.
  - (unobservable) status nodes U
  - health nodes H_S, H_U
- During operation, discrete sensor values and outputs of LTL/MTL formulas are set to S nodes
- Posteriors of the health nodes H_U, H_S reflect the most likely health status of the component



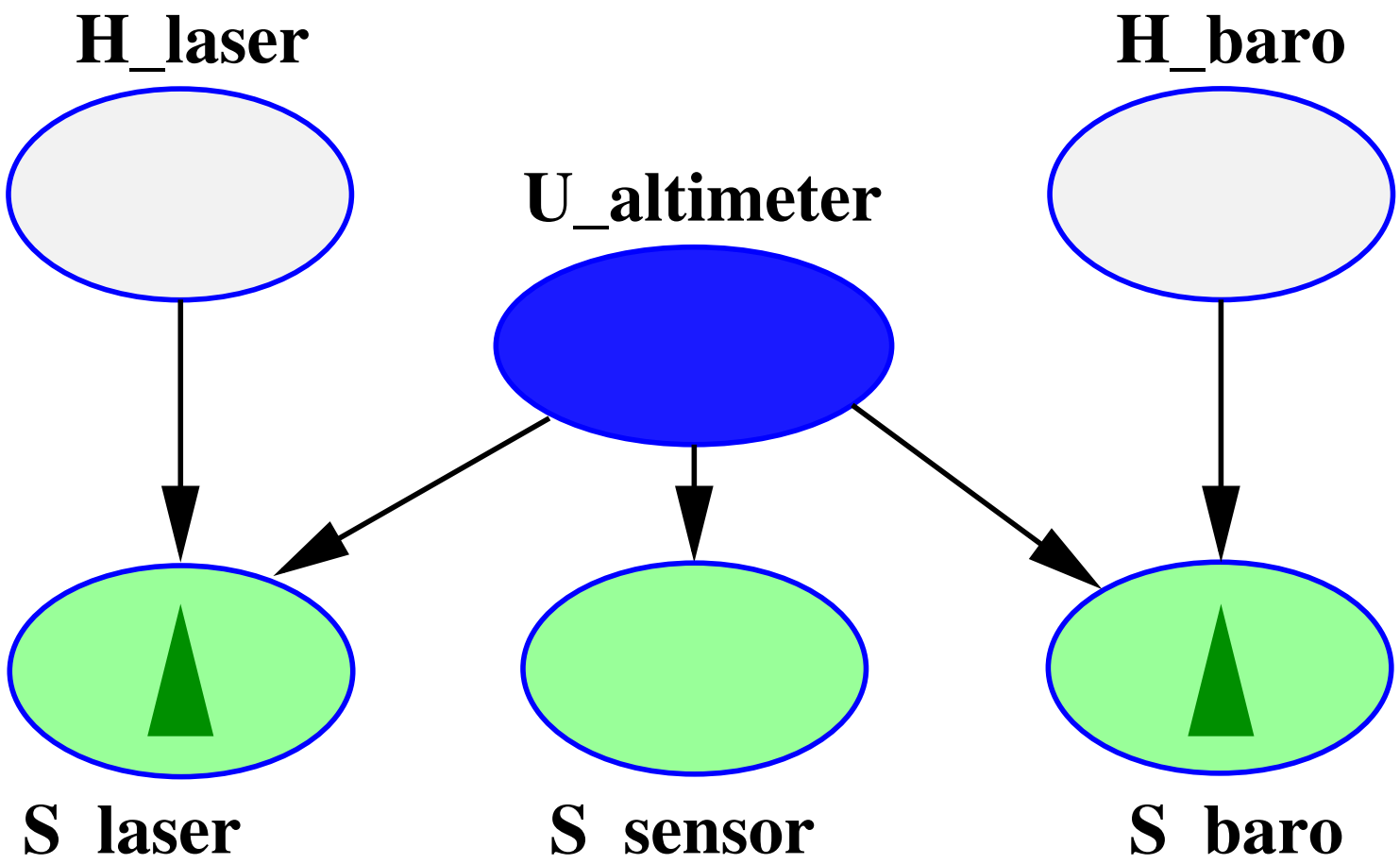
Bayesian Reasoning

- Bayesian models for diagnostic reasoning and health management are well established
- Our Bayesian Networks (BNs) contain
  - (observable) sensor nodes S, which can be noisy of fail.
  - (unobservable) status nodes U
  - health nodes H_S, H_U
- During operation, discrete sensor values and outputs of LTL/MTL formulas are set to S nodes
- Posteriors of the health nodes H_U, H_S reflect the most likely health status of the component

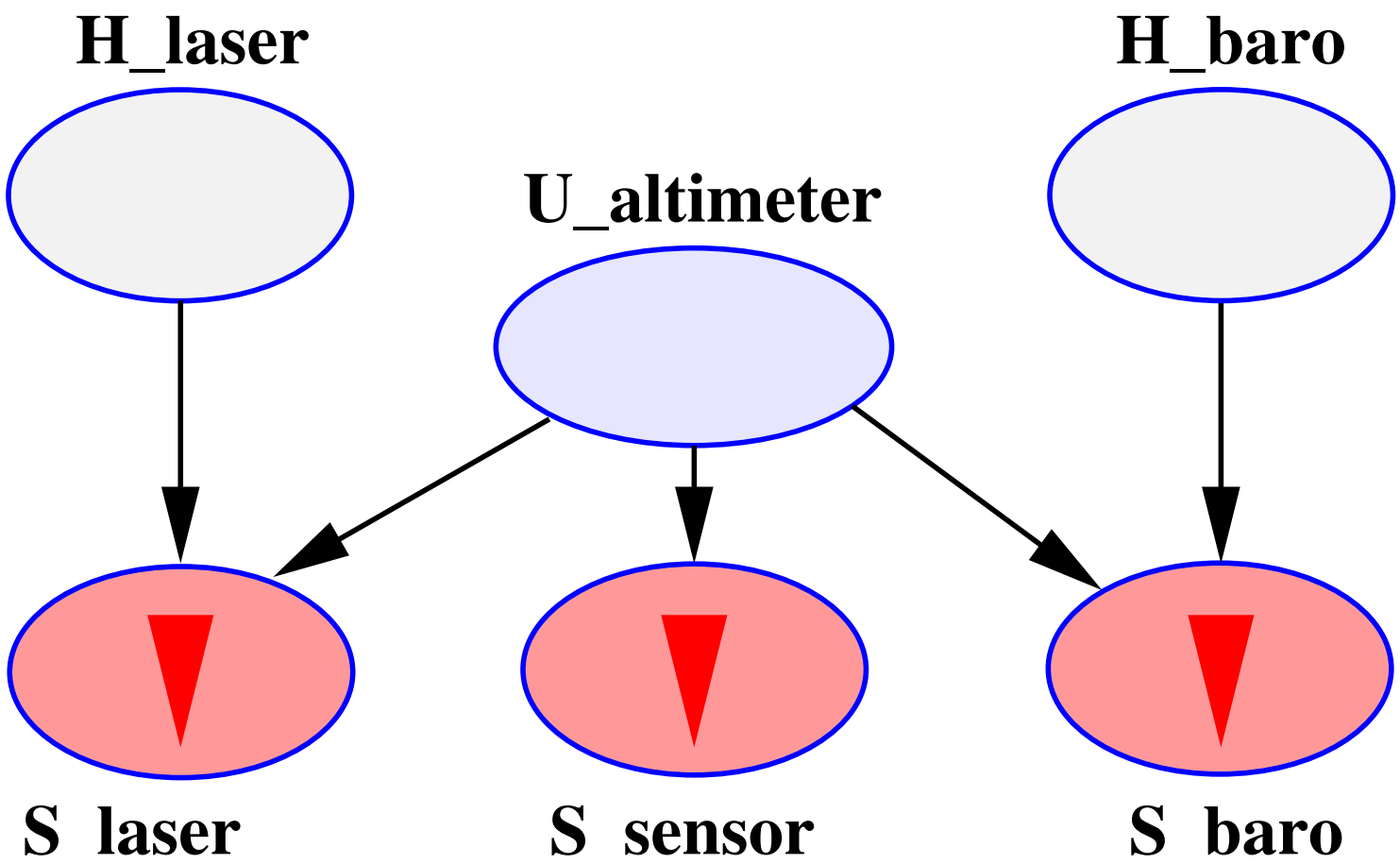


In our framework we do not use Dynamic BNs as temporal aspects are handled by the temporal observers.

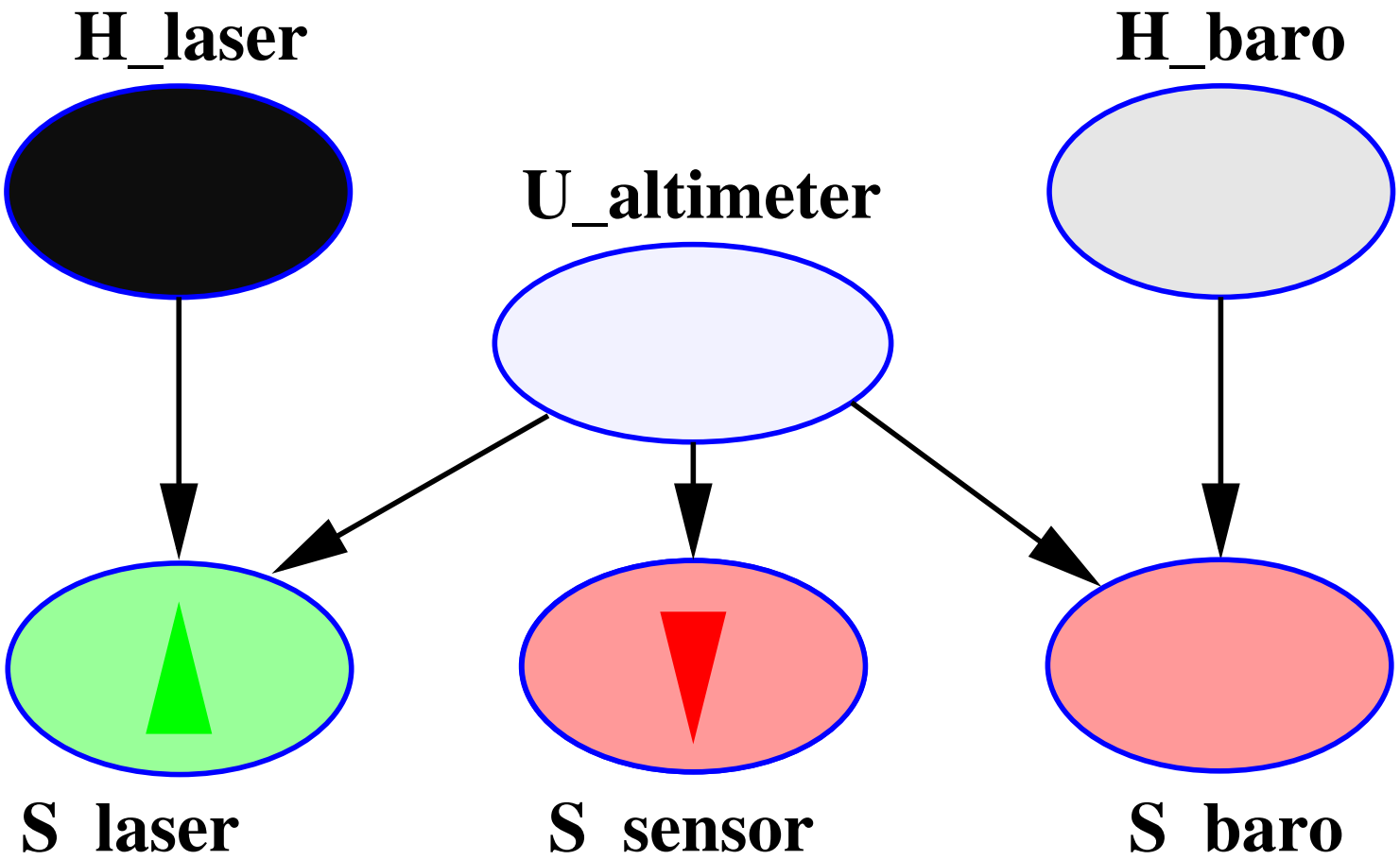
Reasoning about Laser Altimeter Health



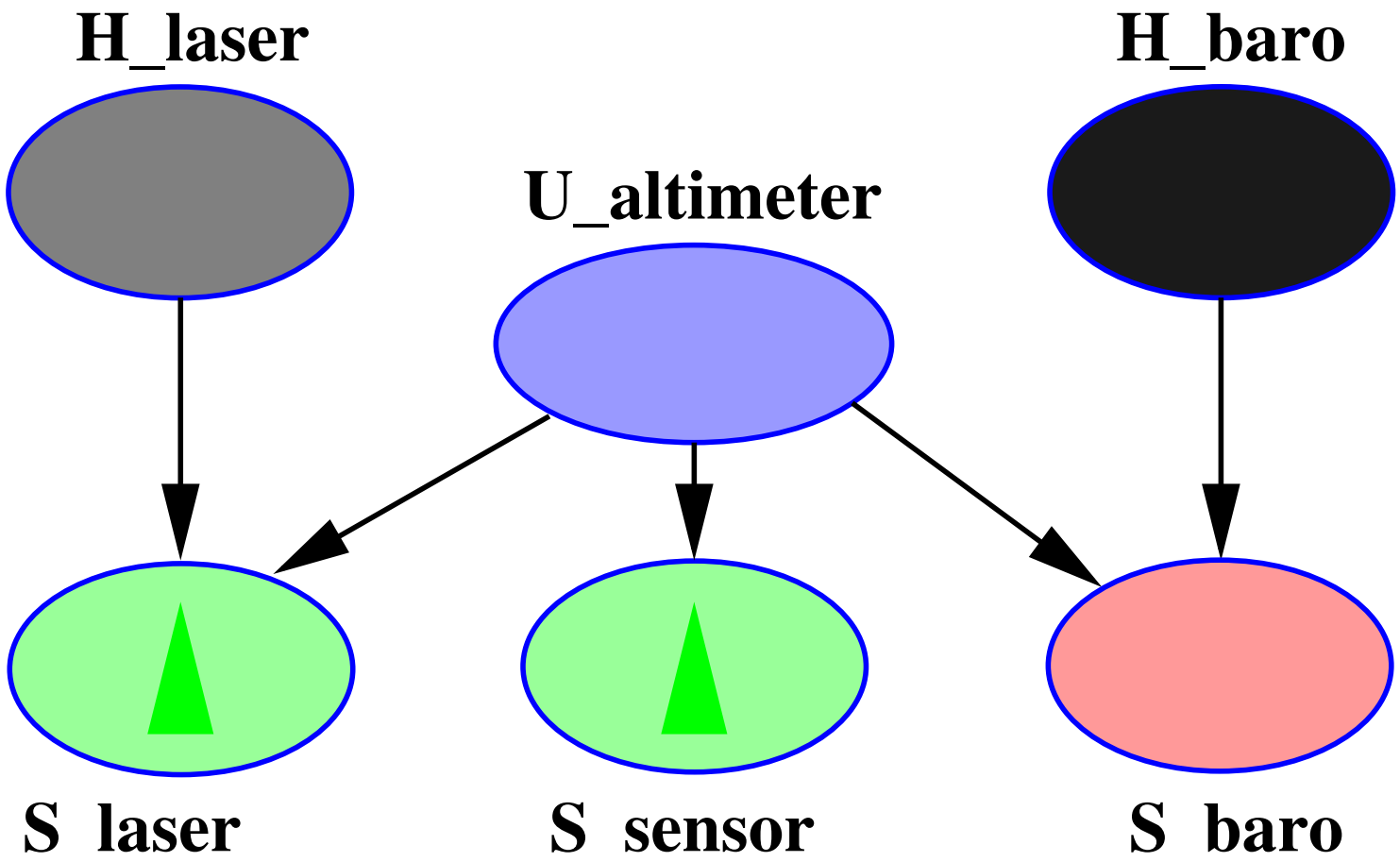
Reasoning about Laser Altimeter Health



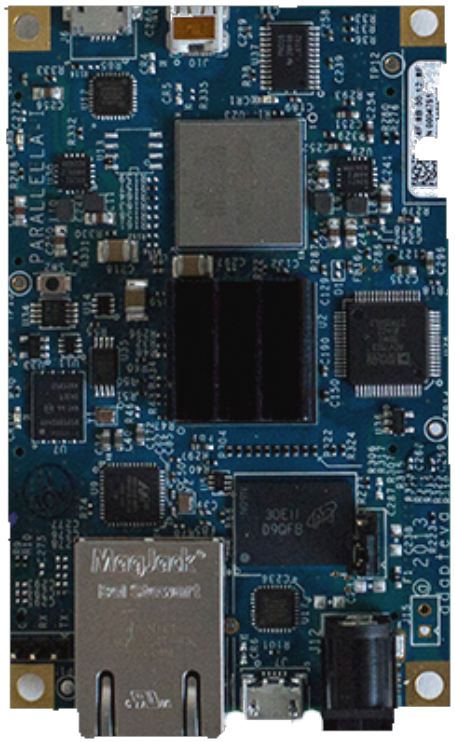
Reasoning about Laser Altimeter Health



Reasoning about Laser Altimeter Health

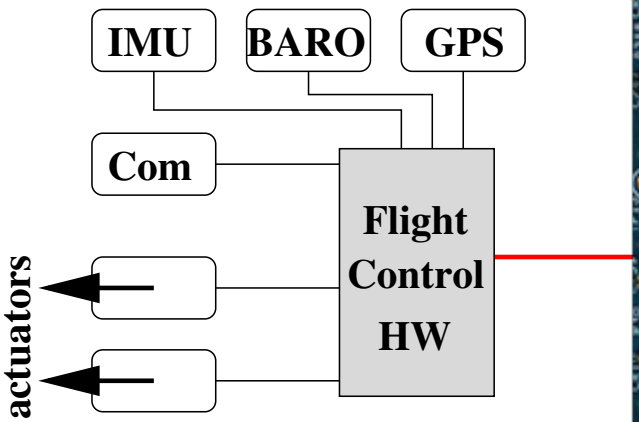


Field-Programmable Gate Array (FPGA)



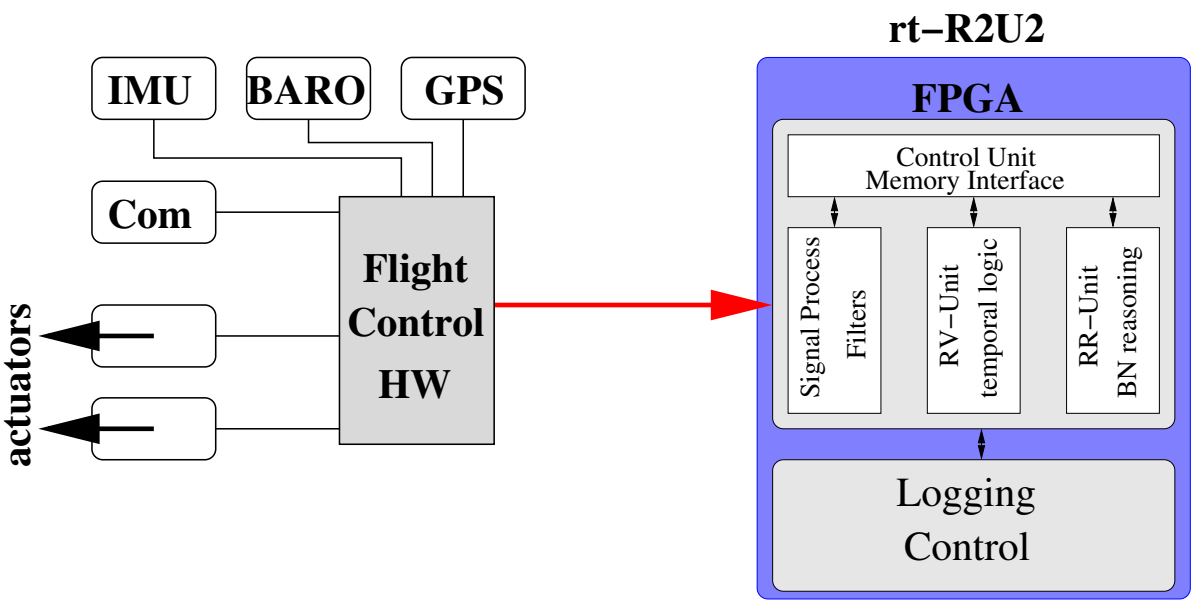
- fast and customizable hardware
- highly parallel architecture
- reconfigurable: program gates interaction
- VHDL (Very High Speed Integrated Circuit Hardware Definition Language)

Hard- and Software Architecture



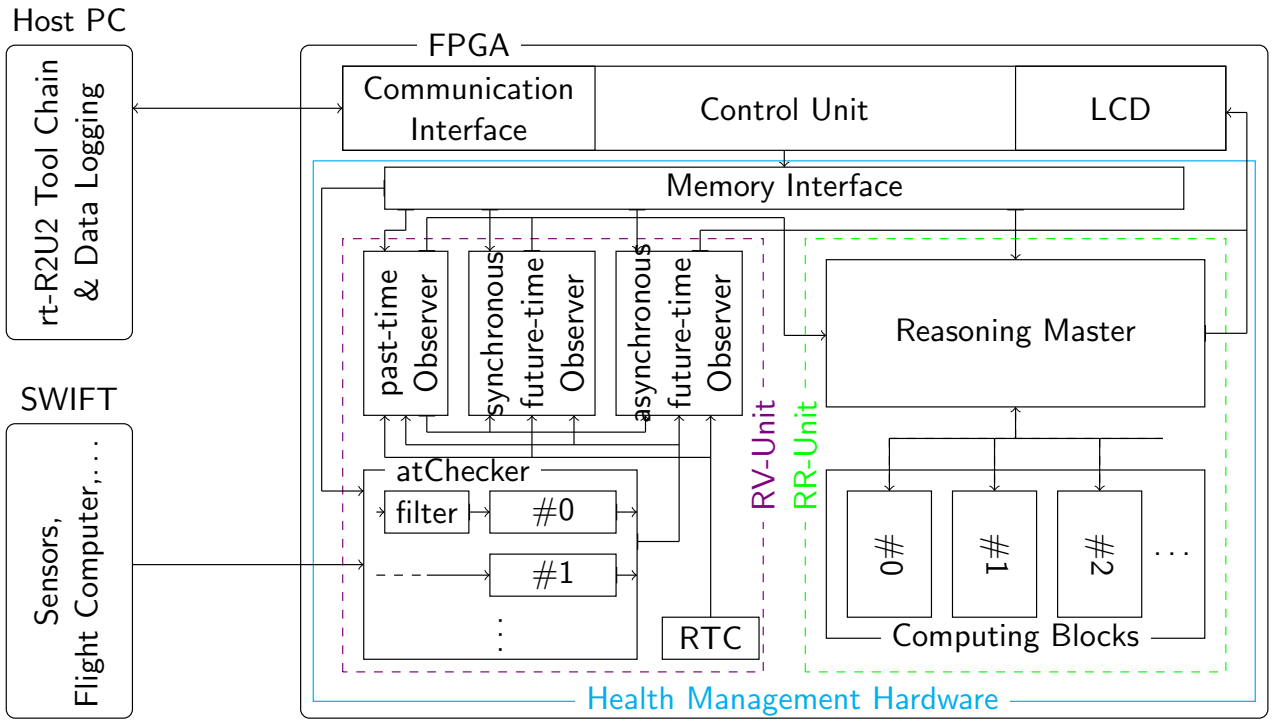
- small rt-R2U2 libaray in flight software to collect relevant variables
- read-only serial (UART) interface to FPGA board
- FPGA for monitoring and reasoning
- data logging/control on Linux system
- For our experiments:
  - Arduino Flight SW
  - Parallella board

Hard- and Software Architecture



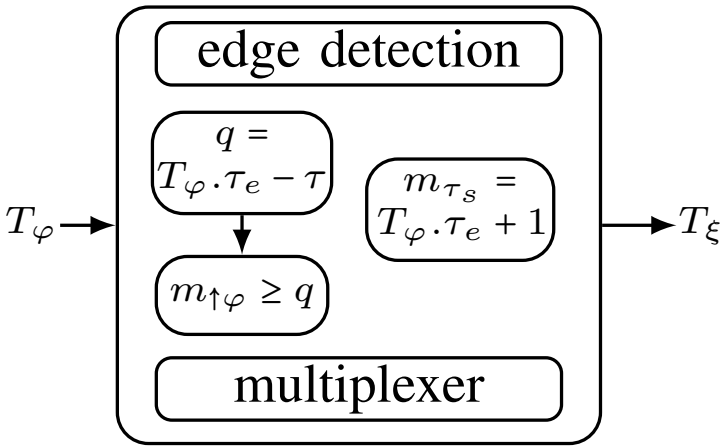
- small rt-R2U2 libaray in flight software to collect relevant variables
- read-only serial (UART) interface to FPGA board
- FPGA for monitoring and reasoning
- data logging/control on Linux system
- For our experiments:
  - Arduino Flight SW
  - Parallella board

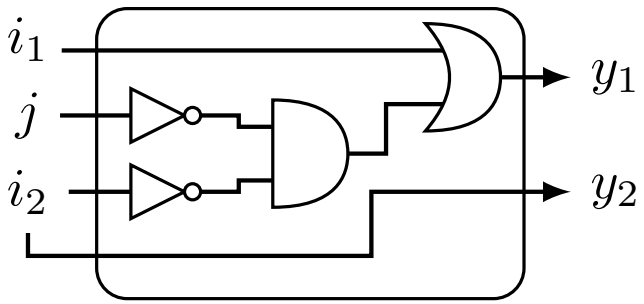
Detailed FPGA Architecture

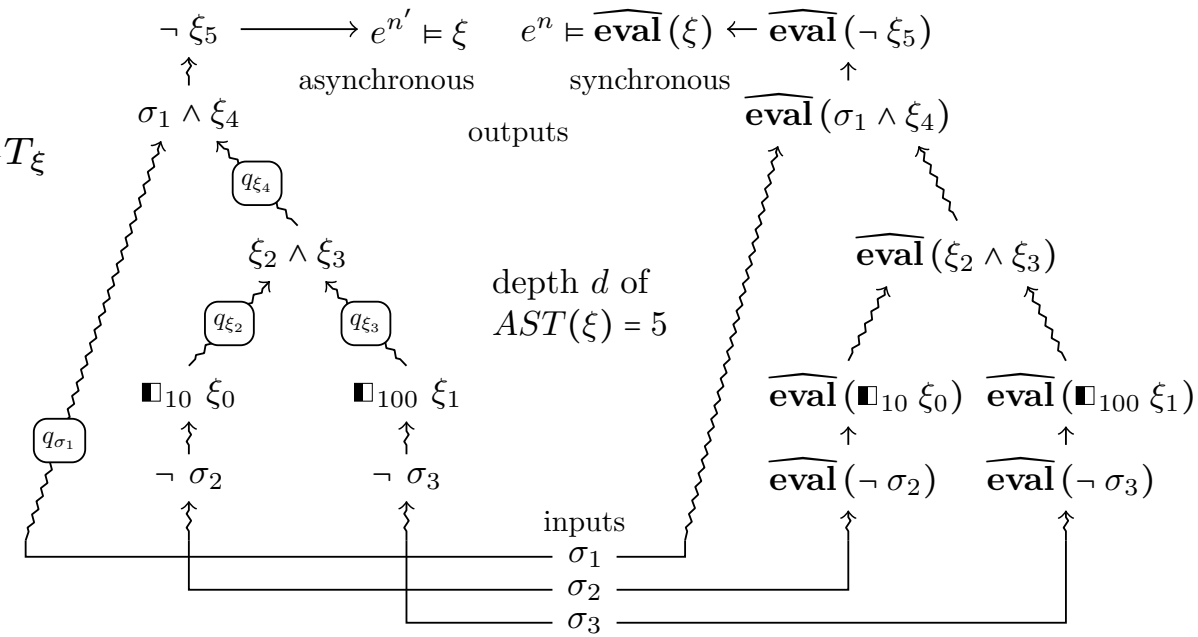


- Customized engines for signal processing, temporal observers, and Bayes reasoning designed in VHDL
- actual model loaded as data, so no new hardware synthesis necessary

FPGA Implementation of Temporal Observers

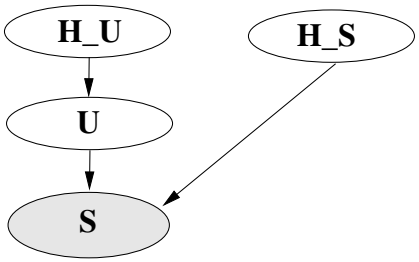






- asynchronous observers: substantial hardware complexity
- synchronous observers: small HW footprint
- parallel processing of sync and async observers

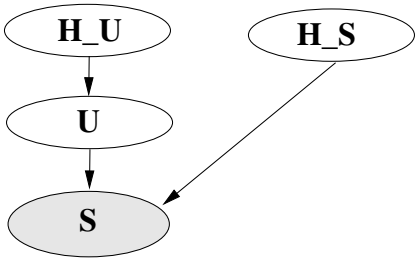
FPGA Implementation of Bayesian networks

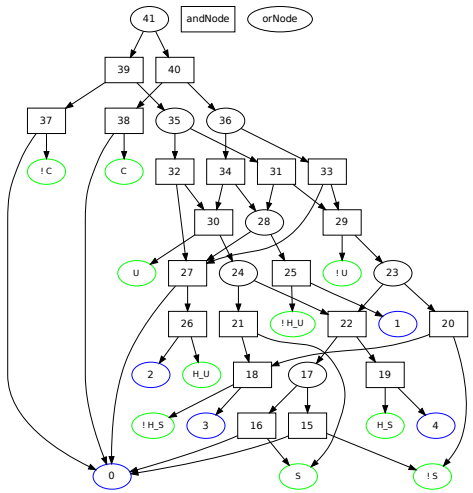


- Bayesian network is translated into an arithmetic circuit (AC)
- AC evaluation
  - only requires addition and muliplication
  - bottom-up and top-down sweep for calculation of posteriors
- fixed-time algorithm, no recursion and suitable for hardware implementation model

FPGA Implementation of Bayesian networks

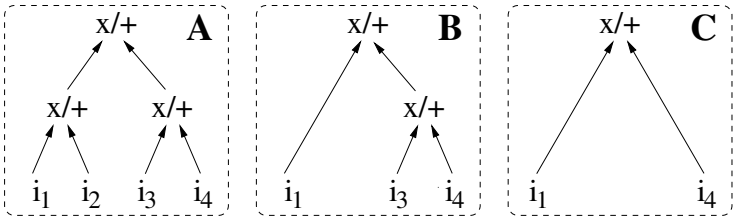
- Bayesian network is translated into an arithmetic circuit (AC)
- AC evaluation
  - only requires addition and muliplication
  - bottom-up and top-down sweep for calculation of posteriors
- fixed-time algorithm, no recursion and suitable for hardware implementation model

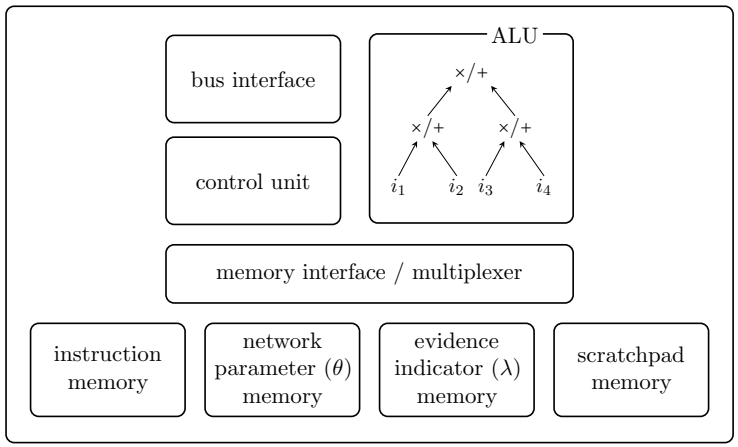




BN Computing Blocks

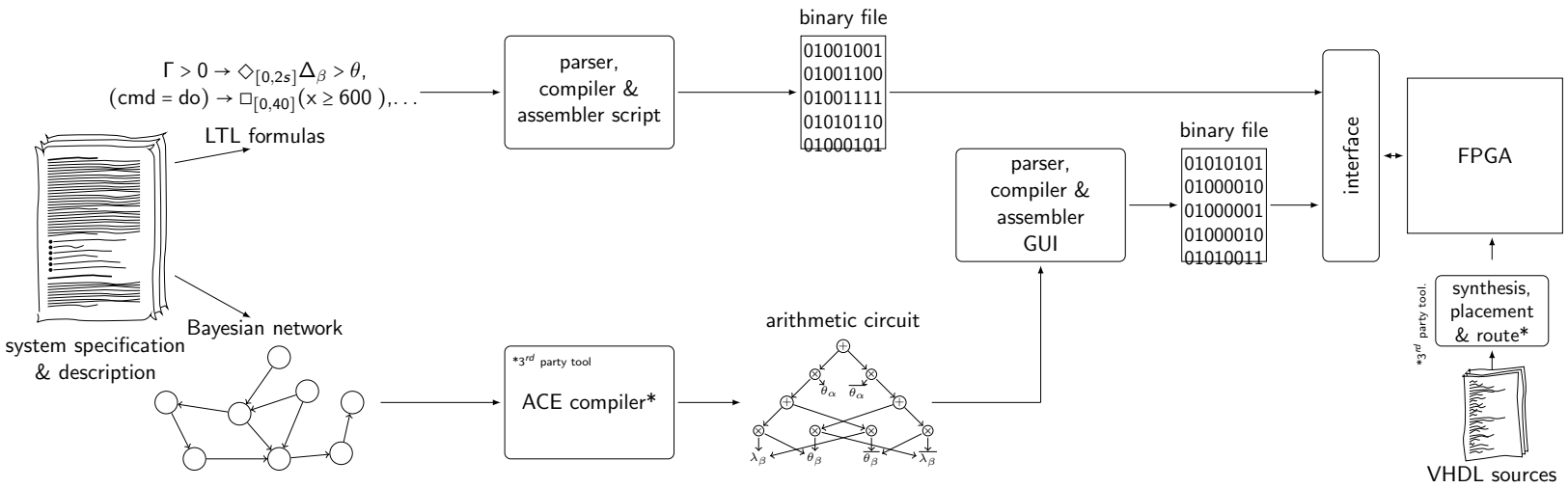
- The entire AC is tiled with 3 types of atomic trees, which are executed by a BN computing block
- Each execution engine contains communication interfaces, local result store, and arithmetic unit for probability calculation
- Posterior calculation is performed by a bottom-up traversal followed by a top-down execution over the AC
- Number of available parallel computing blocks depend on FPGA



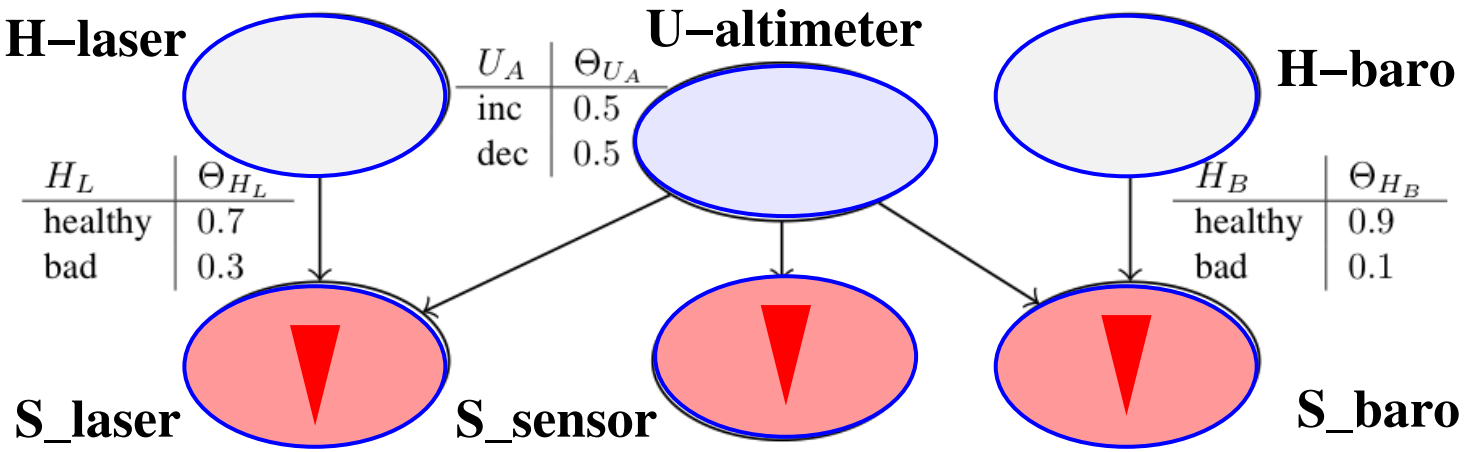


rtR2U2 Tool Chain

- We developed a tool chain to translate SWHM models into efficient FPGA-designs
- Bayesian Network models are compiled into arithmetic circuits that are evaluated by highly parallel special purpose execution units.



Case Study I: Laser Altimeter Failure



H_L	Θ_{H_L}
healthy	0.7
bad	0.3

U_A	Θ_{U_A}
inc	0.5
dec	0.5

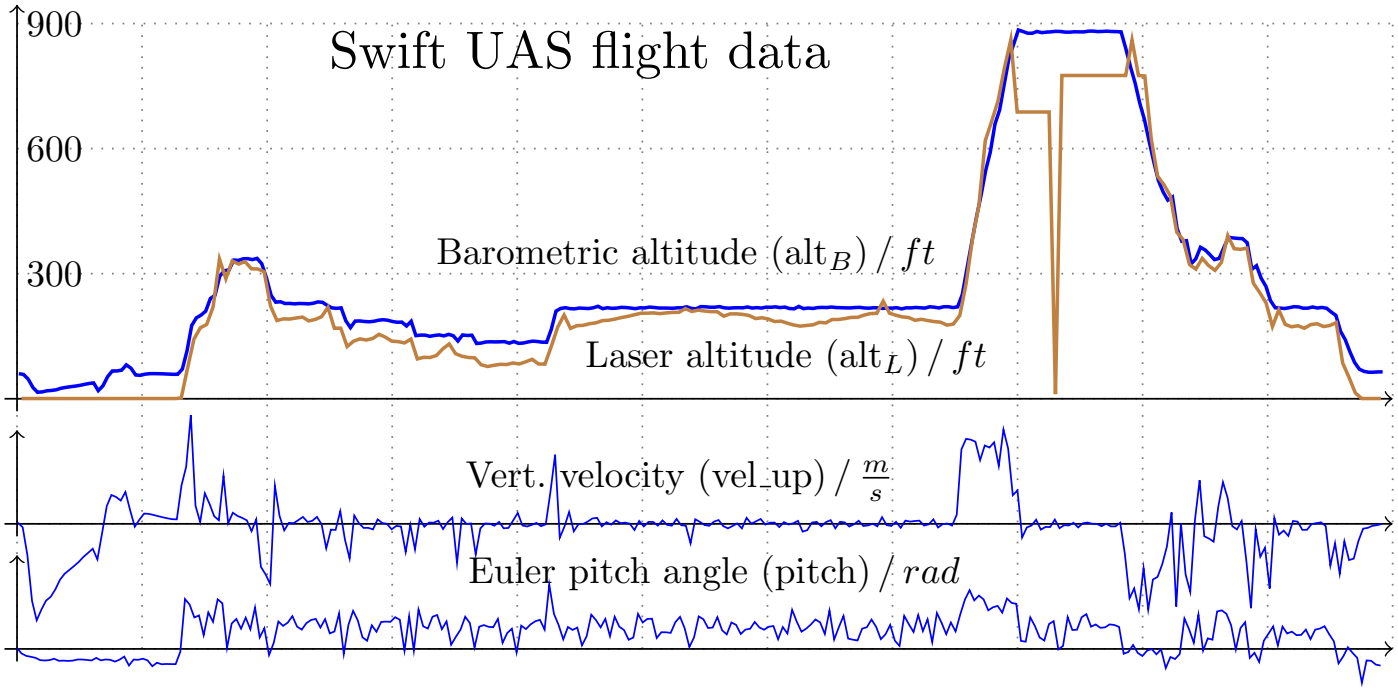
H_B	Θ_{H_B}
healthy	0.9
bad	0.1

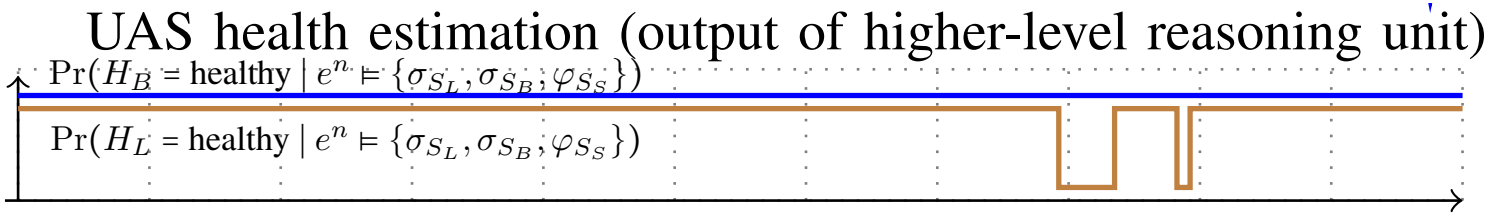
U_A	H_L	S_L	Θ_{S_L}
inc	healthy	inc	1.0
		dec	0.0
	bad	inc	0.5
		dec	0.5
dec	healthy	inc	0.0
		dec	1.0
	bad	inc	0.5
		dec	0.5

U_A	S_S	Θ_{S_S}
inc	inc	0.7
	dec	0.1
	maybe	0.2
dec	inc	0.1
	dec	0.7
	maybe	0.2

U_A	H_B	S_B	Θ_{S_B}
inc	healthy	inc	1.0
		dec	0.0
	bad	inc	0.5
		dec	0.5
dec	healthy	inc	0.0
		dec	1.0
	bad	inc	0.5
		dec	0.5

Case Study I: Laser Altimeter Failure



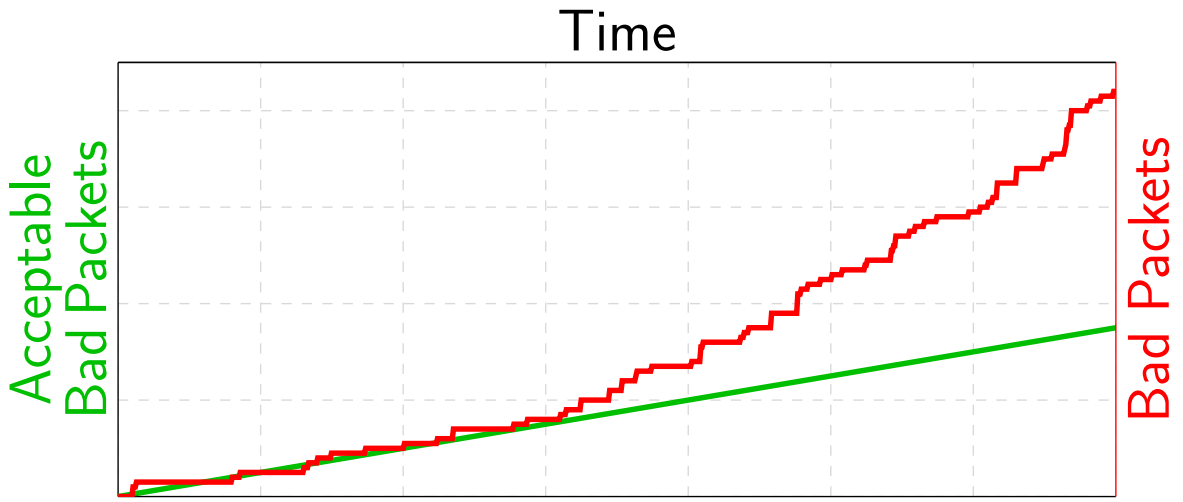


Case study II: Fluxgate Magnetometer Buffer Overflow

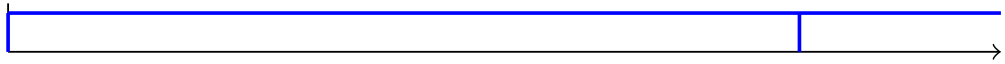
Temporal specification (excerpts):

S_2: **The rate of bad packets N_b^R is low**, no more than one bad packet every 30 seconds.	$\Box_{[0,30]}(N_b^R = 0 \vee (N_b^R \geq 1 \; \mathcal{U}_{[0,30]} N_b^R = 0))$
S_3: **The bad packet rate N_b^R does not appear to be increasing**; we do not see a pattern of three bad packets within a short period of time.	$\neg(\Diamond_{[0,30]} N_b^R \geq 2 \wedge \Diamond_{[0,100]} N_b^R \geq 3)$
S_5: **When aircraft is moving, fluxgate readings should change**. That should not fail more than three times within 100 seconds of each other.	$Eul := (\lvert p\rvert > \theta \vee \lvert q\rvert > \theta \vee \lvert r\rvert > \theta) \rightarrow (\lvert FG_x\rvert > \theta_{FG} \vee \lvert FG_y\rvert > \theta_{FG} \vee \lvert FG_z\rvert > \theta_{FG})$ $\neg(\neg Eul \wedge (\Diamond_{[2,100]}(\neg Eul \wedge \Diamond_{[2,100]} \neg Eul)))$

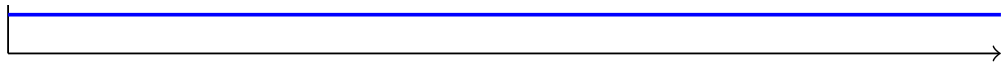
Case study II: Fluxgate Magnetometer Buffer Overflow



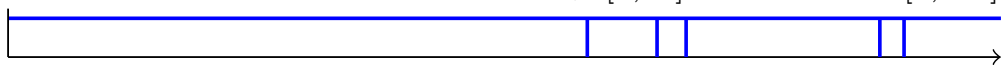
The fluxgate packet transmission rate is appropriate : $63 \leq N_{tot}^R \leq 66$



The number of bad packets is low : $\square_{[0,30]}(N_b^R = 0 \vee (N_b^R \geq 1 \mathcal{U}_{[0,30]} N_b^R = 0))$

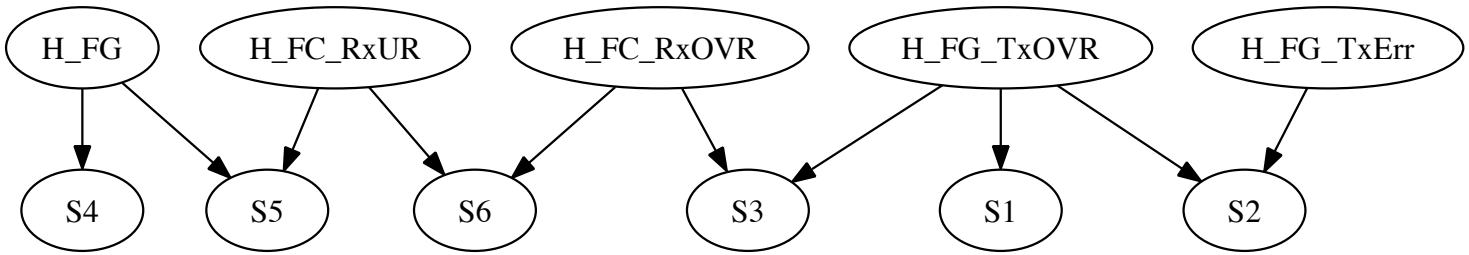


The bad packet rate is not increasing: $\neg(\Diamond_{[0,30]} N_b^R \geq 2 \wedge \Diamond_{[0,100]} N_b^R \geq 3)$

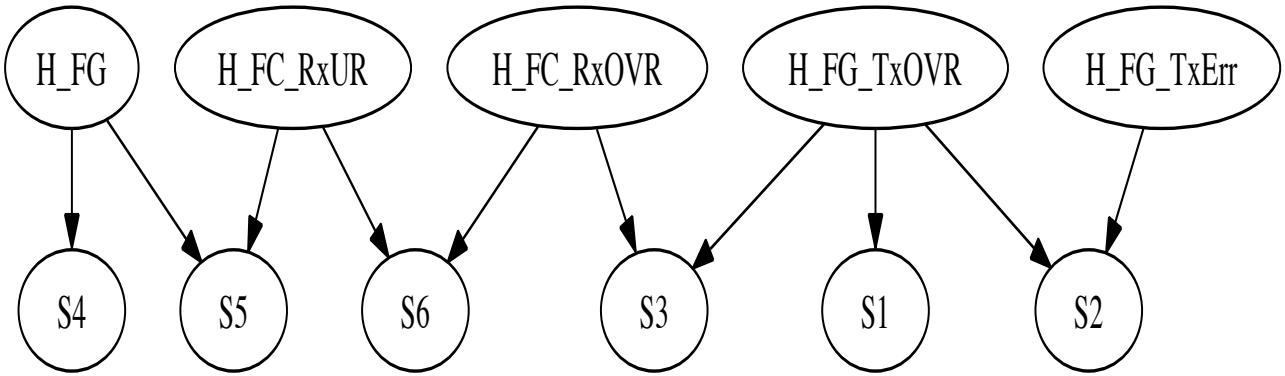


Case study II: Bayesian Health Model

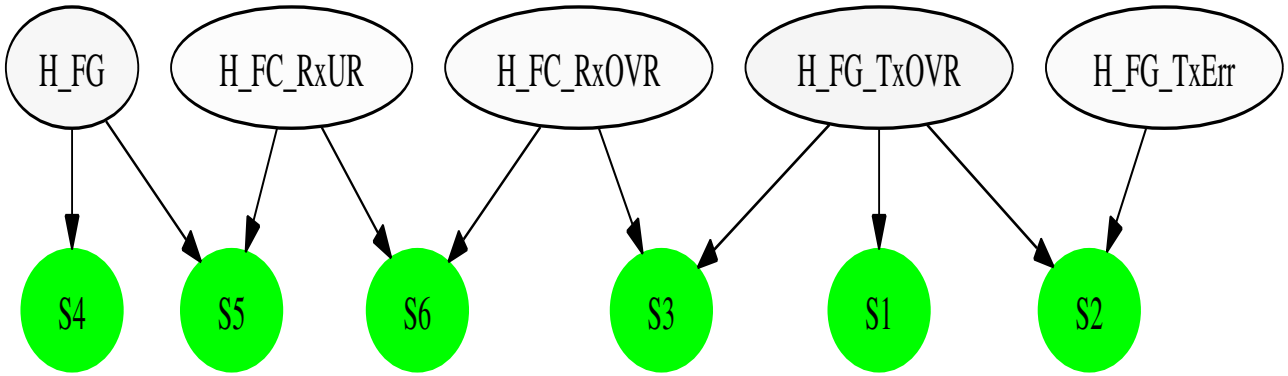
Health Nodes / Failure Modes	
H_FG	**magnetometer sensor**
H_FC_RxUR	Receiver underrun
H_FC_RxOVR	Receiver overrun
H_FG_TxOVR	**Transmitter overrun in sensor**
H_FG_TxErr	Transmitter error in in sensor



Reasoning

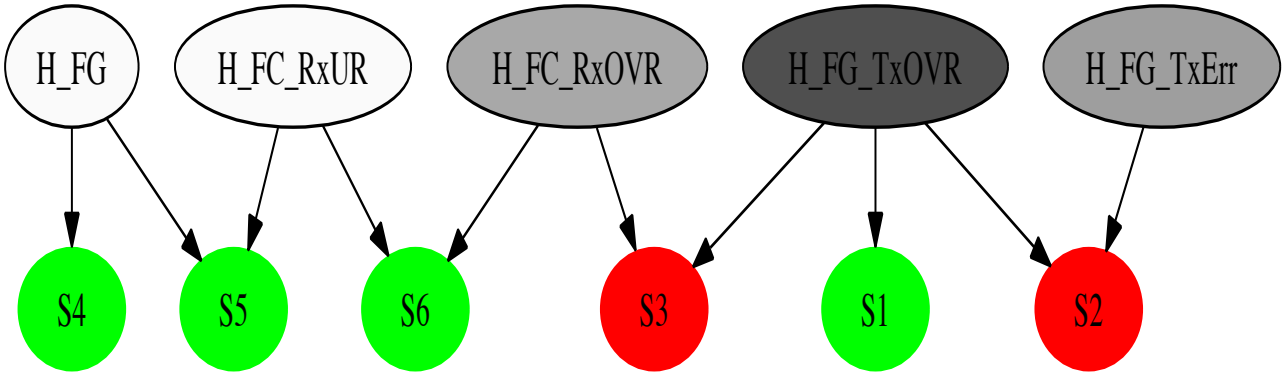


Reasoning



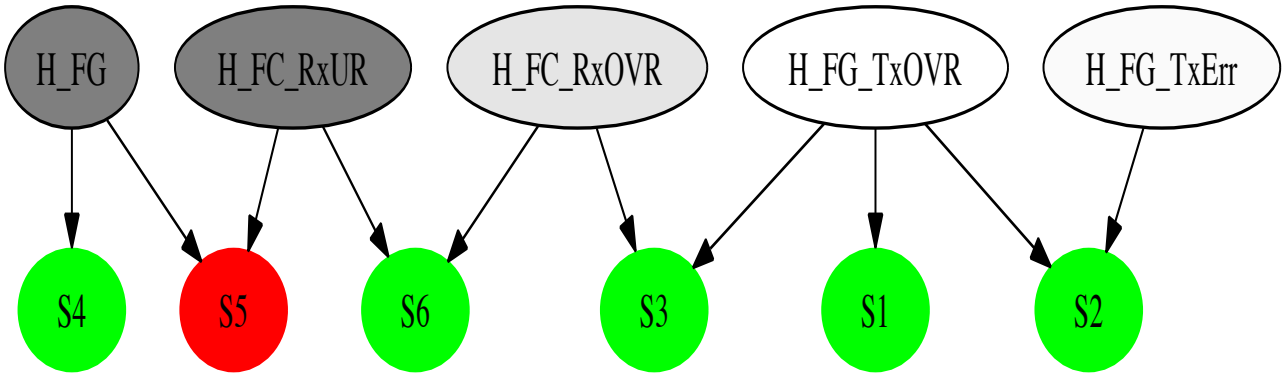
- clamp sensor nodes with results of temporal formulas $S_1, \ldots, S_6$

Reasoning



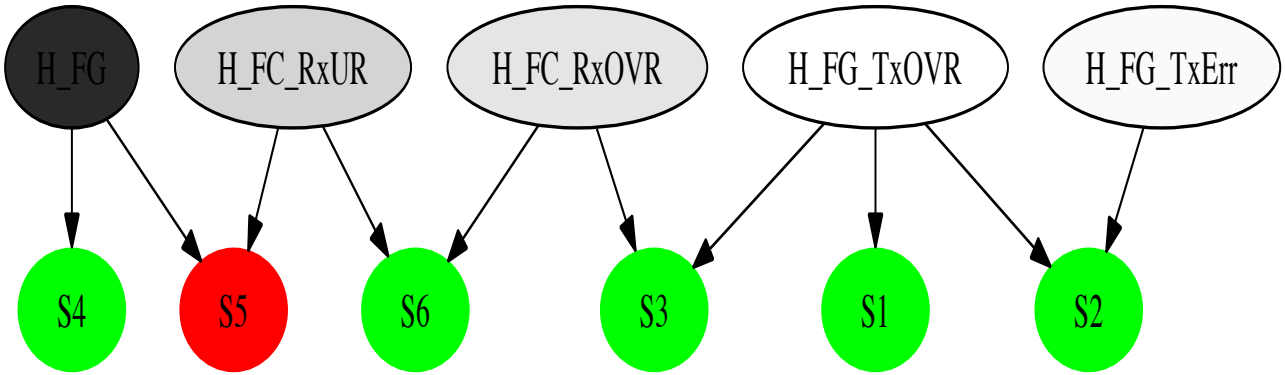
- clamp sensor nodes with results of temporal formulas $S_1, \ldots, S_6$
- S_2 (low error rate) and S_3 (error rate not increasing) needed for diagnosis

Reasoning



- clamp sensor nodes with results of temporal formulas $S_1, \ldots, S_6$
- S_2 (low error rate) and S_3 (error rate not increasing) needed for diagnosis
- BN cannot distinguish between two failure modes

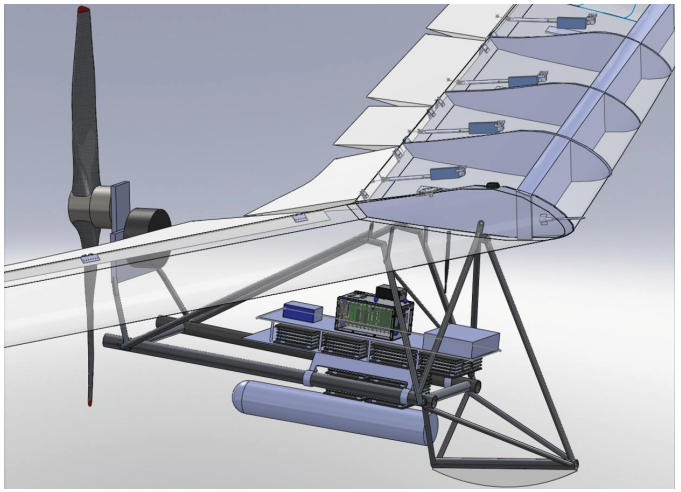
Reasoning



- clamp sensor nodes with results of temporal formulas $S_1, \ldots, S_6$
- S_2 (low error rate) and S_3 (error rate not increasing) needed for diagnosis
- BN cannot distinguish between two failure modes
- Priors in CPT table helps to disambiguate ("Fluxgate not very reliable")

Realization Challenges

- Swift UAS changed to ground test platform
- Switch to easy setup testbed: DragonEyes
- Mounting Parallela board on smaller UAS
- Hardware/software platform limitations
- Clearance for flight tests

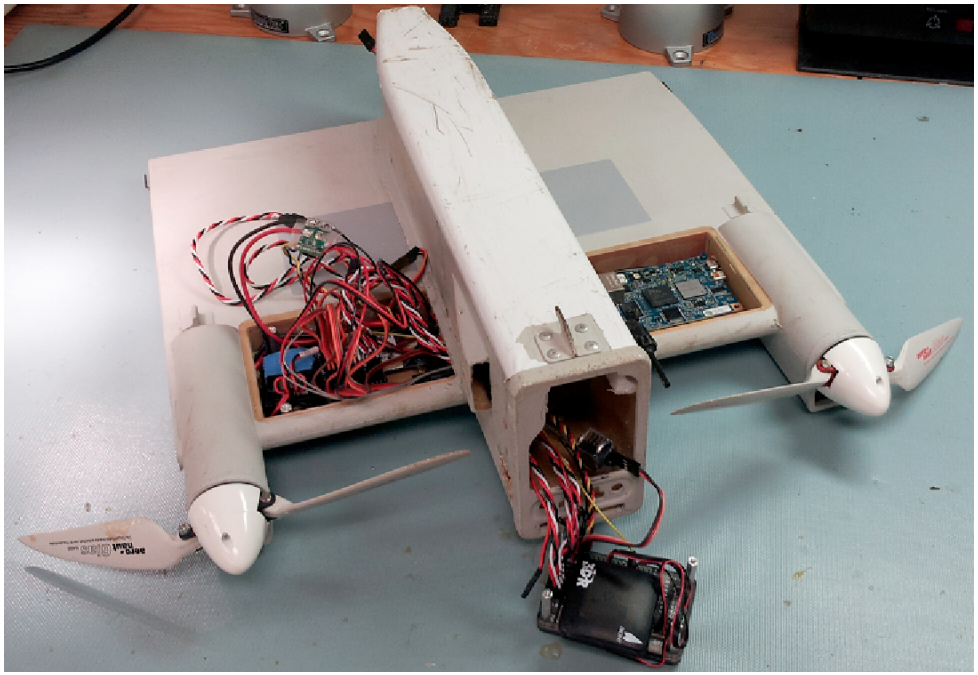


Swift UAS

Swift in Hanger Annex N211-A

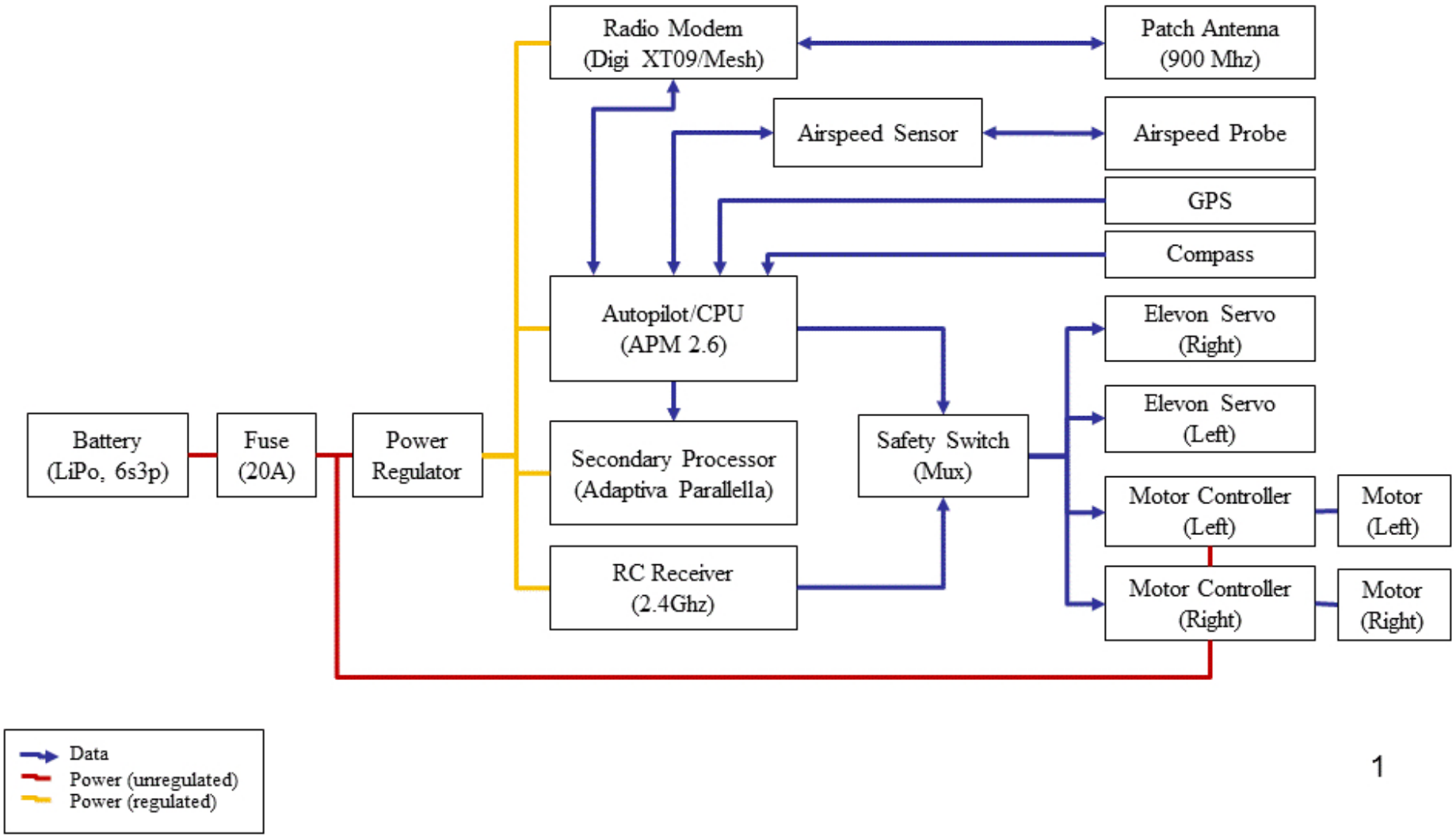
1. all-electric
2. completely autonomous
3. limited weight, power, hardware
4. largely COTS, previously flight-certified components (cheap)
5. developed into a standardized ground test platform for NASA UAS fleet

DragonEye Test Platform

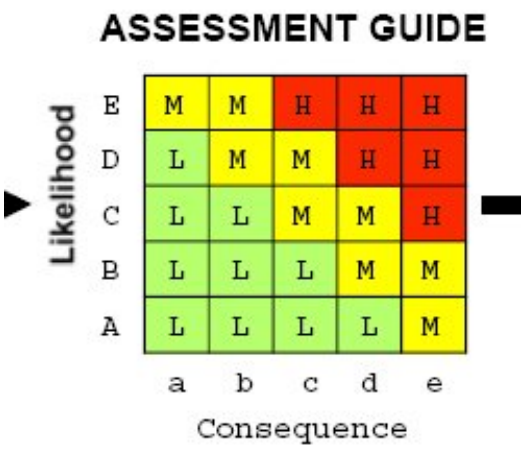


- NASA DragonEye as our test platform
- Open avionics bay. FPGA board (right) w/o heat sink

DragonEye: Current and Future Analysis



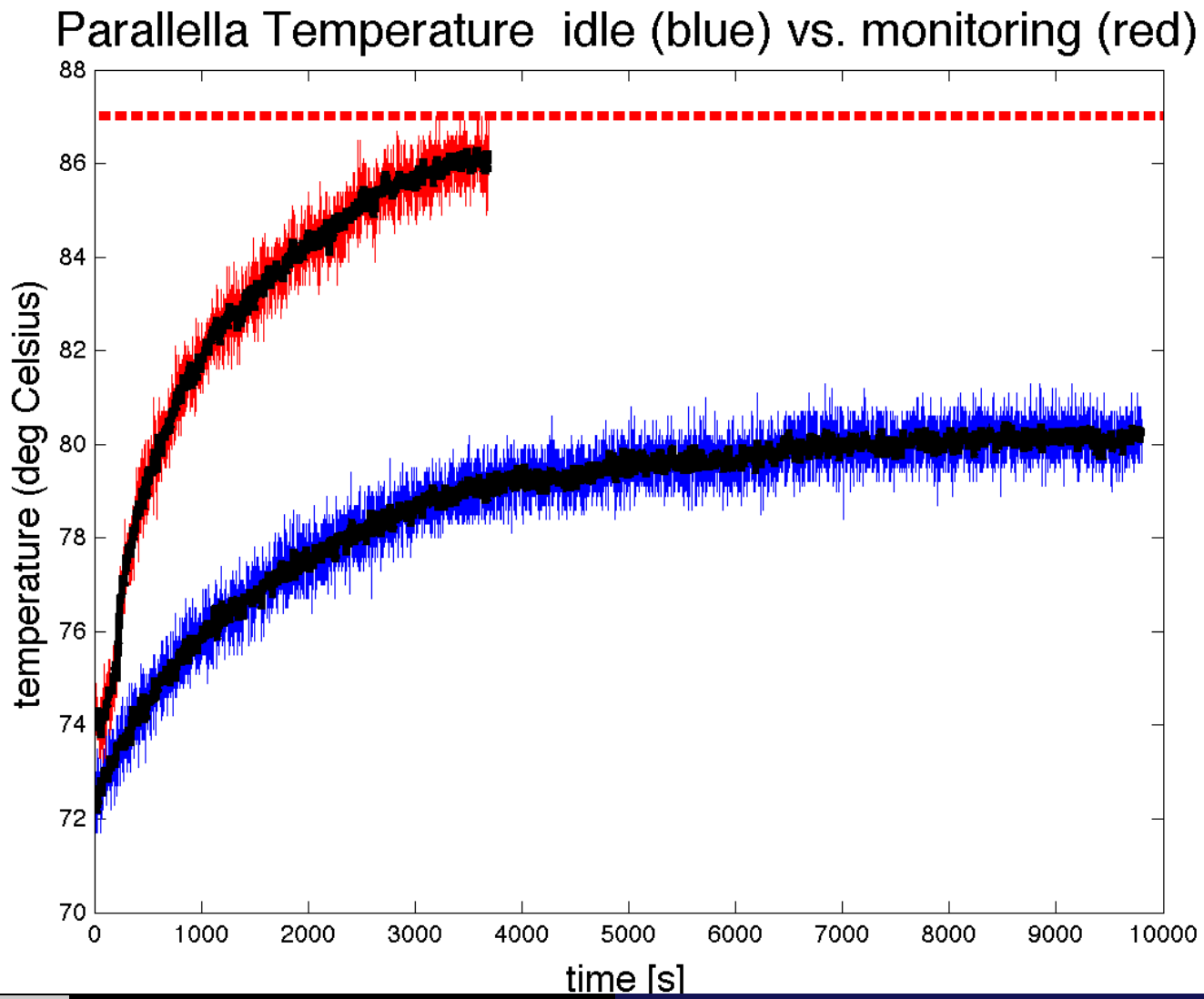
Risk Analysis



likely-hood	consequence a	b	c	d	e
E	--	--	--	--	--
D	--	--	--	--	--
C	6.1 8.2.1 8.2.1	--	--	--	--
B	7.1 8.1 8.2.1 8.2.1 8.2.2 8.2.3 8.2.4 8.2.5 8.2.6 8.2.7	5.4 7.2	--	--	--
A	4.3	5.1 5.2	--	4.1 4.2	--

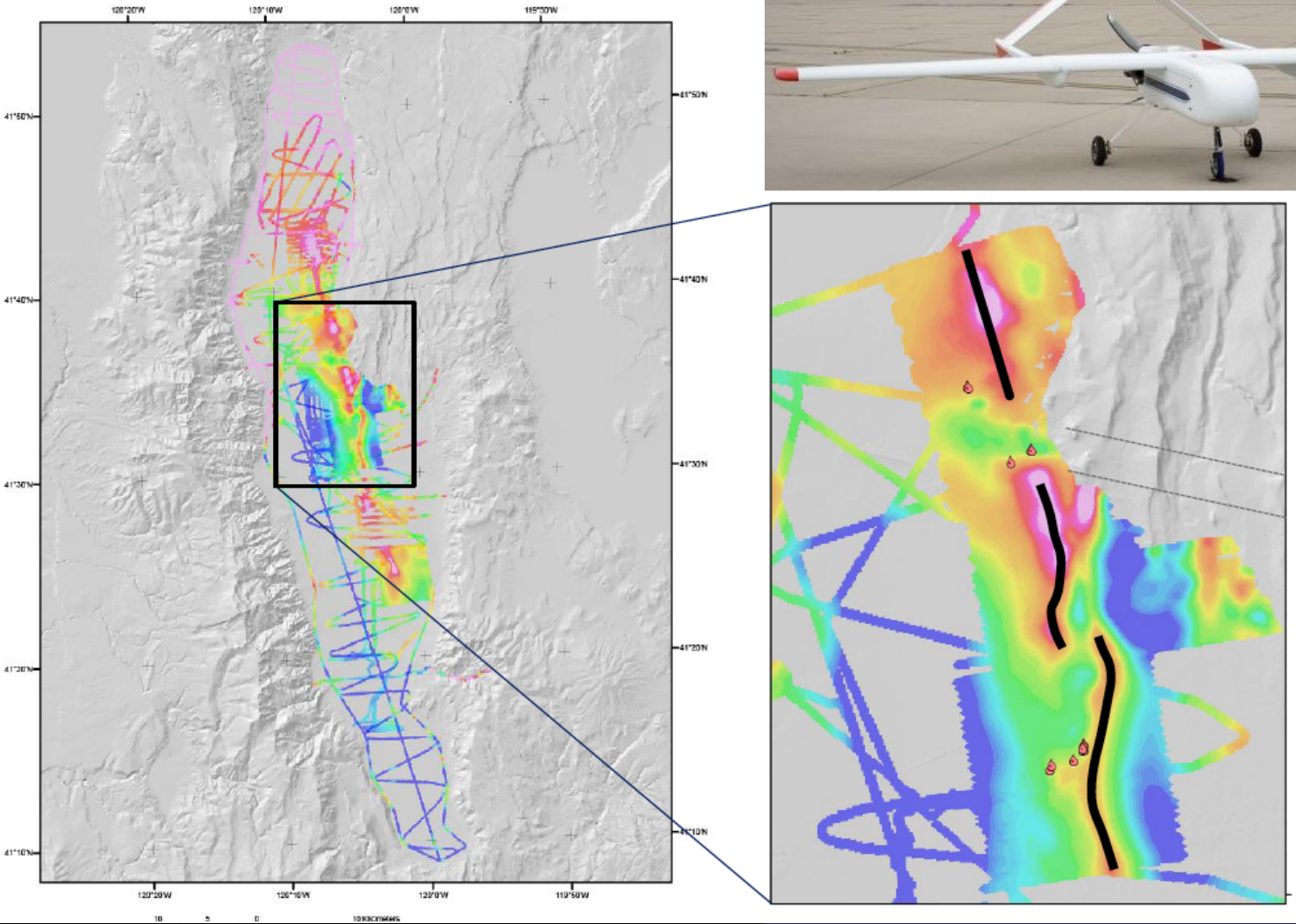
- Detailed risk analysis (23+ page report) carried out for FPGA integration and planned test flights
- Areas:
  - mechanical
  - electrical
  - flight-software
  - interface to FPGA
  - FPGA-board
  - operational
  - test flights with injected failures
- Identified risks could be mitigated and are therefore on lower levels of likelihood and consequence

Temperature Analysis: Parallela Challenged by Overheating



Future Platform: Viking Simulator

Future Missions: Sierra UAS Surveying Earthquakes



Future Platform: Cockpit Operations

Future Work: Summary

- Integrate into other ARMD applications
- Embed, specialize for, and test on other classes of UAS
- Expand to other unmanned vehicles: SmallSAT, rovers, etc.
- Adapt for manned aircraft, cockpit operations

Peer-reviewed Publications & Presentations

Publications:

- Thomas Reinbacher, Kristin Y. Rozier, and Johann Schumann. "Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems." In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 8413 of Lecture Notes in Computer Science (LNCS), pages 357–372, Springer-Verlag, April, 2014.
- Johannes Geist, Kristin Yvonne Rozier, and Johann Schumann. "Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems." In *Runtime Verification (RV14)*, Springer-Verlag, September 22-25, 2014.
- Johann Schumann, Kristin Y. Rozier, Thomas Reinbacher, Ole J. Mengshoel, Timmy Mbaya, and Corey Ippolito. "Towards Real-time, On-board, Hardware-supported Sensor and Software Health Management for Unmanned Aerial Systems." In *International Journal of Prognostics and Health Management (IJPHM)*. (Invited journal paper; To appear)

Presentations:

- K.Y. Rozier: "No More Helicopter Parenting: Intelligent Autonomous Unmanned Aerial Systems." NASA Ames' premier seminar series, the Directors Colloquium, special edition in honor of NASA Ames' 75th Anniversary celebration, by invitation of the Office of the Chief Scientist. NASA Ames Research Center, Moffett Field, California, June 10, 2014.
- J. Schumann: "Towards on-board, hardware-supported Sensor and Software Health Management for UAS." FORTISS, Technische Universität München, Germany, June 2014.

Discussion

- Intelligent UAS capability for self-diagnosis
- Autonomous system health managment with decision-making capability
- Obeying hard requirements for operation: real-time, Realizable, Responsive, Unobtrusive

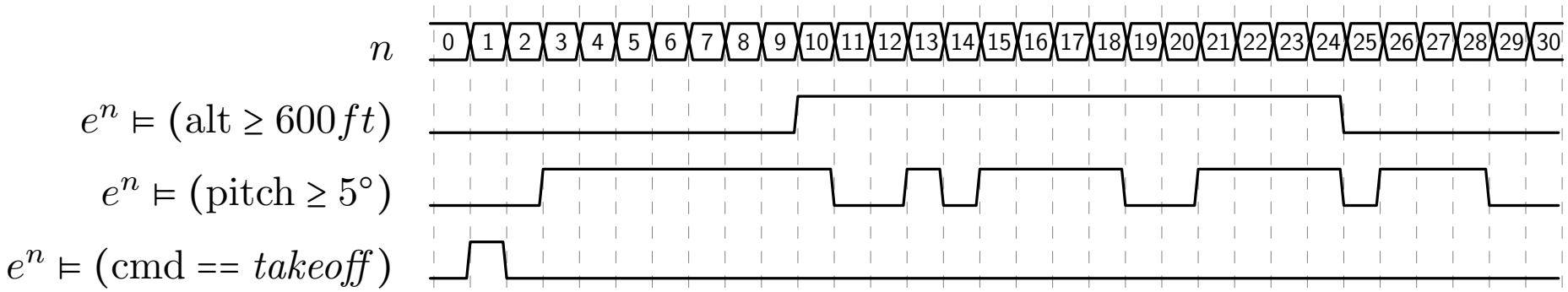
Everything is online! `http://research.kristinrozier.com/R2U2/`

Collaborators:

- Johannes Geist, M.S. Research Intern, University of Applied Sciences Tecnikum Wien, Austria
- Eddy Mazmanian, civil servant NASA Ames Research Center, Moffett Field, CA
- Patrick Moosbrugger, M.S. Research Intern University of Applied Sciences Tecnikum Wien, Austria
- Quoc-Sang Phan, Ph.D. Research Intern Queen Mary University of London, UK
- Thomas Reinbacher, Ph.D. Research Intern Vienna University of Technology, Austria
- Iyal Suresh, sophomore Undergraduate Intern University of California, Los Angeles

BACKUP SLIDES

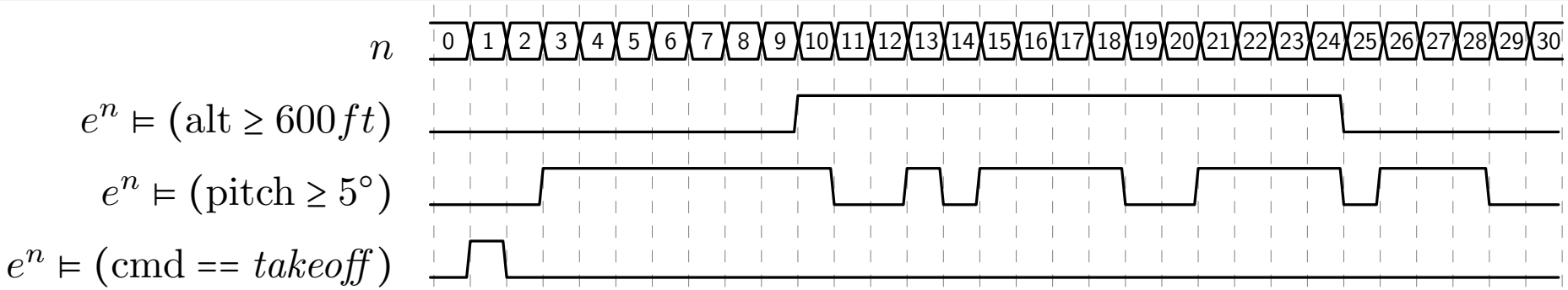
Asynchronous Observers Example



$\blacksquare_5\ (\text{pitch} \geq 5°)$

0	(false,0)	8	(true,3)
1	(false,1)	9	(true,4)
2	(false,2)	10	(true,5)
3	(␣, ␣)	11	(false,11) Resynchronized!
4	(␣, ␣)	12	(false,12)
5	(␣, ␣)	13	(␣, ␣)
6	(␣, ␣)	14	(false,14) Resynchronized!
7	(␣, ␣)	15	(␣, ␣)

Synchronous Observers Example



$$\blacksquare_5\ (\text{alt} \geq 600ft) \wedge (\text{pitch} \geq 5°) \xrightarrow{translate} \xi = \widehat{\textbf{eval}}\,(\widehat{\textbf{eval}}\,(\blacksquare_5\ (\text{alt} \geq 600ft)) \wedge (\text{pitch} \geq 5°))$$

0	(false,0)	8	(false,8)
1	(false,1)	9	(false,9)
2	(false,2)	10	(maybe,10)
3	(false,3)	11	(false,11)
4	(false,4)	12	(false,12)
5	(false,5)	13	(maybe,13)
6	(false,6)	14	(false,14)
7	(false,7)	15	(maybe,15)

Temporal Logic Behavior Properties

Linear Temporal Logic (LTL) formulas reason about linear timelines:

- finite set of atomic propositions {p q}
- Boolean connectives: $\neg$, $\wedge$, $\vee$, and $\rightarrow$
- temporal connectives:

$\mathcal{X}p$	NEXT TIME
$\Box p$	ALWAYS
$\Diamond p$	EVENTUALLY
$p\mathcal{U}q$	UNTIL
$p\mathcal{R}q$	RELEASE